SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO–MATEMATIČKI FAKULTET
MATEMATIČKI ODSJEK

Domina Hozjan

BLOCKCHAIN

Diplomski rad

Voditelj rada:
izv.prof.dr.sc. Luka Grubišić

Zagreb, veljača, 2017.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Zahvaljujem svom mentoru izv.prof.dr.sc. Luki Grubišiću na strpljenju, pomoći i vodstvu pri izradi ovog diplomskog rada.

Želim se zahvaliti i svim prijateljicama i prijateljima, posebno Marti, Atini i Kristini i bratu Luki na ispijenim kavama, zajedničkim učenjima i ludim provodima. Hvala im što su bili tu za mene i kad sam bila pod stresom i kad se sa mnom nije dalo razgovarati i što su mi svojim prisustvom uljepšali i obogatili studentske dane.

Hvala baki Ljubici i djedu Đuri čije su mi riječi uvijek bile motivacija. Hvala im na tome što je njihov ponos na svoju unuku bio nešto zbog čega se uvijek vrijedno boriti.

Na kraju najveću zahvalu želim izreći svojoj mami Milici, koja je uvijek bila uz mene, pružala mi podršku i bez koje sve ovo što sam dosad postigla ne bi bilo moguće. Od prvog dana mog školovanja zajedno sa mnom nestrpljivo iščekivala rezultate svakog ispita i radovala se mojim uspjesima. Hvala joj što je uvijek vjerovala u mene.

Sadržaj

Uvod

Blockchain tehnologija omogućava da se digitalna informacija distribuira između svih čvorova koji sudjeluju u sustavu. Tako svaki čvor održava svoju kopiju svake relevantne informacije i nema potrebe za središnjim autoritetom koji kontrolira informacije. Kontrola je također distribuirana, pomoću mehanizama za validaciju svaki čvor može biti siguran da je informacija zapisana na blockchain točna. Ova tehnologija nastala je za potrebe digitalne valute Bitcoin, koja je predstavljena 2008. godine a s radom je započela godinu kasnije (2009. godine).

U suvremenom, informacijskom društvu informacije imaju veliku važnost. Količina (povjerljivih) informacija koju razmjenjujemo narasla je značajno, no način na koji se one razmjenjuju i pohranjuju nije se mnogo promijenio od 90-ih godina prošlog stoljeća. Ako se radi o informacijama čiji integritet želimo zaštiti tada od sustava za pohranu traži se da ima sljedeće karakteristike:

- Informacije se kreću kroz sigurnu mrežu.
- Informacije se ne smiju modificirati tokom ni nakon zapisivanja.
- Pravo korištenja digitalnih ili materijalnih dobara vezanih za informaciju ima samo ovlašteni korisnik.
- Brzina dijeljenja informacija mora biti što veća moguća.
- Pregled informacija se može jednostavno izvršiti od strane bilo kojeg zainteresiranog korisnika.

Blockchain pruža maksimalnu zaštitu integriteta zapisa korištenjem kriptografskih metoda. Zapisi su distribuirani, svaki čvor u sustavu posjeduje ekvivalentne podatke što je postignuto algoritmima za postizanje konsenzusa od kojih su najpoznatiji proof-of-work i proof-of-stake algoritam. Zapise nije moguće mijenjati ili ih ometati. Blockchain je sastavljen od blokova međusobno povezanih u lanac. Svaki blok sadrži niz zapisa. Blokovi se povezuju algoritmom koji koristi kriptografsku

hash funkciju. Veza između blokova se ne može krivotvoriti osim ako napadač ima ogromne računalne resurse na raspolaganju.

U dosadašnjem poslovanju svaka zainteresirana strana u postupku održava lokalne baze podataka s različitim informacijama. Bilo da se radi o centraliziranoj ustanovi poput banke ili nekoj organizaciji u sustavu poslovanja. Podaci se dijele između stranaka samo kada ima potrebe za tim. Integritet podataka u lokalnim bazama organizacije pod svaku cijenu žele zaštititi, no napadi na takve digitalne podatke sve su češći. Napad na izoliranu lokalnu bazu podataka može imati katastrofalne posljedice. Tada postoji mogućnost rekreiranja baze iz backupa pri čemu dolazi do gubitka dijela informacija. Nepotpunost podataka može uzrokovati i značajne financijske gubitke. Ako napad nije odmah otkriven u budućnosti podaci neće biti konzistentni s podacima partnera u poslovanju. To će narušiti međusobno povjerenje, a otklanjanje problema uzrokovanih napadom će biti dug i mukotrpan proces. Blockchain tehnologija nam omogućava da podaci postanu korisniji ne samo jednoj organizaciji, nego svim partnerima u mreži. S druge strane, podacima kao i ostalim procesima između partnera se upravlja na kontroliran način uz visoku razinu sigurnosti i povjerenja.

Poglavlje 1

SHA-256 kriptografska hash funkcija

1.1 Uvod

Da bismo mogli proučavati *blockchain* kao strukturu podataka i *blockchain* kao tehnologiju najprije moramo definirati što su to kriptografske hash funkcije. Ovo poglavlje opisuje osnovne karakteristike kriptografskih hash funkcija i algoritam funkcije SHA-256. Kao što ćemo vidjeti u poglavljima koja slijede u sklopu *blockchain tehnologije* funkcija SHA-256 je između ostalog korištena pri povezivanju blokova, dodavanju novih blokova, generiranju sažetaka od svih podataka iz jednog bloka.

1.2 Kriptografska hash funkcija

Kriptografska hash funkcija je posebna klasa hash funkcije koja ima određena svojstva koja je čine prikladnom za uporabu u kriptografiji. Općenito, hash funkcija je bilo funkcija koja za ulaz ima podatke proizvoljne veličine, a kao izlaz vraća podatke fiksne veličine. Vrijednost hash funkcije često naziva hash vrijednost ili kratko *hash* dok se ulazni podatak naziva poruka. Kriptografske hash funkcije su jednosmjerne odnosno nemaju inverz. Jedini način da se kreiraju ulazni podaci kriptografske hash funkcije iz izlaza je pokušati pretraživanje brute-force algoritmom. Isprobavanjem svih mogućih vrijednosti ulaza kako bi se vidjelo koji od ulaza odgovara izlazu koji posjedujemo.

Poželjno je da kriptografska hash funkcija zadovoljava sljedećih 5 svojstava:

- Deterministička je. Vrijedi, ako su dva izlaza dobivena pomoću iste funkcije različiti, tada su i ulazi bili različiti.
- Lako i brzo se može izračunati vrijednost funkcije za bilo koji ulaz.

- Neisplativo je generirati ulaz za određeni izlaz, isprobavanjem svih mogućih vrijednosti ulaza.
- Mala promjena na ulaznim podacima treba promijeniti vrijednost funkcije, tako da se ne naslućuje nikakva sličnost između stare i nove vrijednosti funkcije.
- Postoji mogućnost kolizije, dobivanja istih vrijednosti za različite ulazne podatke, ali je neisplativo traženje dvaju takvih ulaznih podataka.

Kriptografske hash funkcije imaju mnogo primjena. Mogu se koristiti za implementaciju različitih struktura podataka poput tablica, lista ili stabala. Primjenjuju se za digitalno potpisivanje poruka između korisnika u nekom nesigurnom sustavu. Također pri radu s datotekama hash funkcije omogućuju stvaranje digitalnog "otiska prsta" (*eng. fingerprint*) na sadržaj datoteke. Pomoću "otiska prsta" lako je identificirati je li sadržaj datoteke promijenjen. Mnogi operacijski sustavi koriste kriptografske hash funkcije za enkripciju zaporki te su one sastavni dio mnogih mehanizama za provjeru autentičnosti. Kriptovalute (*eng. cryptocurrencies*) takve funkcije koriste kako bi bez središnjeg autoriteta postigle siguran prijenos novca.

1.3 SHA-256

Slijedi opis SHA-256 funkcije. Prije samog djelovanja funkcije potrebno je ulazni podatak odnosno poruku proširiti tako da ukupna duljina poruke u bitovima bude djeljiva s 512. Označimo ukupnu duljinu poruke P s d, proširivanje se provodi na sljedeći način:

1. Dodajemo bit "1" na kraj poruke.
2. k puta dodajemo bit "0", gdje je k najmanje pozitivni rješenje jednadžbe $d + 1 + k \equiv 448 \bmod 512$
3. Na kraju dodajemo blok od 64 bita koji predstavlja broj d u binarnom zapisu.

Pogledajmo poruku "bok" u 8-bitnoj ASCII notaciji. Binarni prikaz te poruke je sljedeći:

`01100010 01101111 01101011.`

Duljina binarnog prikaza d iznosi 24, što nije djeljivo s 512 pa provodimo prethodno navedeni algoritam kako bi proširili poruku. Iz $24 + 1 + k \equiv 448 \bmod 512$ dobivamo da je k=423. Dok je binarni zapis broja 24 jednak 11000. Tako dobivamo proširenu poruku:

$$\texttt{01100010 01101111 01101011 1}\ \underbrace{00...0}_{423}\ \underbrace{00...00011000}_{64}$$

Nakon proširenja, poruku P je potrebno podijeliti na blokove $P^{(1)}, P^{(2)}, ..., P^{(N)}$ svaki veličine 512 bitova. Za svaki $i = 1, 2, ..., N$ s $P_0^{(i)}$ označimo prvih 32 bita bloka $P^{(i)}$ ukupne duljine 512 bitova, s $P_1^{(i)}$ sljedećih 32 bita i tako sve do $P_{15}^{(i)}$.

Djelovanje funkcije SHA-256 možemo opisati rekurzivnom formulom:

$$H^{(i)} = H^{(i-1)} + KOM_{P^{(i)}}(H^{(i-1)}) \text{ za } i = 1, 2, ..., N \tag{1.1}$$

gdje je hash $H^{(i)}$ dobiven u svakom međukoraku duljine 256 bita, a KOM predstavlja SHA-256 kompresijsku funkciju koju ćemo opisati u nastavku. Operator $+$ odnosi se na zbrajanje riječi modulo 2^{32}. $H^{(N)}$ je traženi rezultat, odnosno hash poruke P. Početna vrijednost rekurzije (1.1) broj $H^{(0)}$ prikazujemo kao niz brojeva u heksadecimalnom zapisu koji predstavljaju prvih 32 bita decimalnog dijela drugog korijena prvih 8 prostih brojeva:

$$H_1^{(0)} = \texttt{6a09e667}$$

$$H_2^{(0)} = \texttt{bb67ae85}$$

$$H_3^{(0)} = \texttt{3c6ef372}$$

$$H_4^{(0)} = \texttt{a54ff53a}$$

$$H_5^{(0)} = \texttt{510e527f}$$

$$H_6^{(0)} = \texttt{9b05688c}$$

$$H_7^{(0)} = \texttt{1f83d9ab}$$

$$H_8^{(0)} = \texttt{5be0cd19}.$$

Kompresijska funkcija djeluje nad blokovima poruke duljine 512 bitova i 256 bitnim hash vrijednostima dobivenim u međukoracima rekurzije (1.1). Prije samog pseudokoda opišimo pomoćne funkcije i vrijednosti. Slijede definicije 6 pomoćnih funkcija koje se koriste u SHA-256 algoritmu:

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\Sigma_0(x) = S^2(x) \oplus S^{13}(x) \oplus S^{22}(x)$$
$$\Sigma_1(x) = S^6(x) \oplus S^{11}(x) \oplus S^{25}(x)$$
$$\sigma_0(x) = S^7(x) \oplus S^{18}(x) \oplus R^3(x)$$
$$\sigma_1(x) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x).$$

Tablica 1.1 daje pregled i opis operatora korištenih u navedenim funkcijama. Ulazni argumenti i izlazne vrijednosti tih operatora su izrazi u binarnom zapisu duljine 32 bita.

Tablica 1.1: Pregled operatora

Oznaka	Tip	Opis
$\wedge$	Binarni	Na mjestu i ima vrijednost bita 1 onda i samo onda kada su bitovi iz obiju varijabli na mjestu i jednaki 1, u protivnom 0.
$\neg$	Unarni	Na mjestu i ima vrijednost bita 1 onda i samo onda kada je bit ulazne varijable na mjestu i jednaki 0, u protivnom 0.
$\oplus$	Binarni	Na mjestu i ima vrijednost bita 1 onda i samo onda kada jedna od varijabli na mjestu i ima vrijednost bita 1, u protivnom 0.
$+$	Binarni	Zbrajanje modulo 2^{32}
R^n	Unarni	Pomak u desno za n bitova
S^n	Unarni	Rotacija u desno za n bitova

Označimo s $B_0, B_1, ..., B_{63}$ blokove poruka dobivene sljedećom formulom, za fiksni $i = 1, ..., N$, gdje je N broj blokova veličine 512 bita proširene poruke P:

$$B_j = \begin{cases} P_j^{(i)}, & \text{za } j = 0, 1, ..., 15 \\ \sigma_1(B_{j-2}) + B_{j-7} + \sigma_0(B_{j-15}) + B_{j-16}, & \text{inače} \end{cases}$$

Pomoćne heksadecimalne vrijednosti $K_0, K_1, ..., K_{63}$ korištene u pseudokodu algoritma predstavljaju prvih 32 bita decimalnog dijela trećeg korijena prvih 64 prostih brojeva i iznose:

```
428a2f98 71374491 b5c0fbcf e9b5dba5 3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3 72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc 2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7 c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13 650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3 d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208 90befffa a4506ceb bef9a3f7 c67178f2.
```

Konačno, pseudokod algoritma SHA-256 izgleda ovako:

Za $i = 1, ..., N$, gdje je N broj blokova veličine 512 bita proširene poruke P
{

•Registre a, b, c, d, e, f, g, h inicijaliziraj hash vrijednostima $(i-1)$-og međukoraka iz (1.1):

$$a \leftarrow H_1^{(i-1)}$$

$$b \leftarrow H_2^{(i-1)}$$

$$\vdots$$

$$h \leftarrow H_8^{(i-1)}.$$

•Računanje funkcije $KOM_{P^{(i)}}(H^{(i-1)})$ iz (1.1):
Za $j = 0, ..., 63$
{

Izračunaj vrijednosti $Ch(e, f, g), Maj(a, b, c), \Sigma_0(a), \Sigma_1(e)$ i B_j

$$T_1 \leftarrow h + \Sigma_1(e) + Ch(e, f, g) + K_j + B_j$$

$$T_2 \leftarrow \Sigma_0(a) + Maj(a, b, c)$$

$$h \leftarrow g$$

$$g \leftarrow f$$

$$f \leftarrow e$$

$$e \leftarrow d + T_1$$

$$d \leftarrow c$$

$$c \leftarrow b$$

$$b \leftarrow a$$

$$a \leftarrow T_1 + T_2$$

}

•Računanje $i-te$ hash vrijednosti:

$$H_1^{(i)} \leftarrow a + H_1^{(i-1)}$$

$$H_2^{(i)} \leftarrow b + H_2^{(i-1)}$$

$$\vdots$$

$$H_8^{(i)} \leftarrow h + H_8^{(i-1)}.$$

}

Na kraju, $H^{(N)} = (H_1^{(N)}, H_2^{(N)}, H_3^{(N)}, H_4^{(N)}, H_5^{(N)}, H_6^{(N)}, H_7^{(N)}, H_8^{(N)})$ je hash vrijednost poruke P.

Poglavlje 2

Blockchain tehnologija

2.1 Uvod

Blockchain (lanac blokova) predstavlja distribuiranu strukturu podataka odnosno listu digitalnih informacija podijeljenu između svih čvorova koji sudjeluju u sustavu. U ovom poglavlju opisat ćemo značajke *blockchain tehnologije* i način na koji tehnologija funkcionira. Također je ćemo opisati strukturu jednog bloka u lancu i način povezivanja tih blokova. Kao primjer prikazat ćemo strukturu bloka kakva se primjenjuje u *blockchainu* digitalne kriptovalute `Bitcoin` koja je ujedno i prvi sustav u kojem se *blockchain* koristi. Ostale javne i privatne implementacije *blockchaina* koriste jednaku ili vrlo sličnu strukturu bloka. Shodno `Bitcoin` protokolu objašnjen je način funkcioniranja *blockchaina* na primjeru zapisa transakcija u blokove dok kod ostalih aplikacija možemo naići na doista različite formate i značenje digitalnih podataka koji su pohranjeni u *blockchainu*.

2.2 Što je blockchain?

Blockchain tehnologija nastala je za potrebe digitalne valute `Bitcoin`, no kasnije su potencijal te tehnologije prepoznale mnoge industrije, naročito financijski sektor. `Bitcoin` je korištenjem *blockchaina* i kriptografskih funkcija postigao sigurne transakcije digitalnog novca bez središnjeg autoriteta (banke). Ovdje *blockchain* igra ulogu glavne knjige u kojoj je zapisana svaka transakcija ikad izvršena u `Bitcoin sustavu`.

Neke od osnovnih značajki *blockchaina* su:

- Uobičajeno je da je sustav koji koristi *blockchain* izgrađen prema modelu ravnopravnih partnera (peer-to-peer).
- Sustav je u potpunosti decentraliziran, nema potrebe za središnjim autoritetom.
- Svaki novi zapis je u gotovo realnom vremenu distribuiran između mnoštva čvorova.
- U svrhu identifikacije sudionika u sustavu, potvrde identiteta, dokazivanja autentičnosti i u nekim slučajevima iskorištavanja prava za čitanje/pisanje koristi se kriptografija.
- Čvorovi sustava mogu dodavati podatke u *blockchain*.
- Čvorovi sustava mogu čitati podatke iz *blockchaina*.
- *Blockchain* ima razvijen mehanizam koji onemogućuje promjenu nad podacima koji su jednom upisani u *blockchain* ili u najmanju ruku omogućuje lako otkrivanje promjena na podacima.

2.3 Struktura bloka

Blockchain se kao što mu samo ime govori sastoji od blokova. Blok je struktura podataka u kojoj su zapisane digitalne informacije koje se dijele putem *blockchaina*. Iz tablice 2.1 vidimo da se jedan blok sastoji od zaglavlja u kojem su upisani meta-podaci te liste digitalnih informacija varijabilne dužine.

Tablica 2.1: Struktura bloka

Veličina	Naziv	Opis
4 bajta	Veličina bloka	Veličina bloka u bajtovima
80 bajtova	Zaglavlje bloka	Meta-podaci o bloku
1-9 bajtova	Brojač zapisa	Koliko zapisa sadrži blok
Varijabilno	Zapisi	Zapisi pohranjeni u bloku

2.3.1 Zaglavlje bloka

Zaglavlje svakog bloka sastoji se od 80 bajtova podataka koji služe kao dodatne tehničke informacije o bloku i povezivanju blokova u lanac. Struktura zaglavlja bloka dana je u tablici 2.2.

Tablica 2.2: Struktura zaglavlja bloka

Veličina	Naziv	Opis
4 bajta	Verzija	Verzija protokola u vrijeme nastajanja bloka (Specifično za Bitcoin)
32 bajta	Hash prethodnog bloka	Referenca na prethodni blok u lancu koji još nazivamo roditelj bloka
32 bajta	Korijen binarnog hash stabla	Kriptografski hash koji sadrži informacije o svim zapisima u bloku
4 bajta	Vremenska oznaka	Vrijeme kada je blok kreiran i uključen u *blockchain*
4 bajta	Težinska oznaka	Težina algoritma čije je rješenje potrebno za uključivanje bloka u *blockchain*
4 bajta	*Nonce*	Broj pomoću kojeg je riješen algoritam za uključivanje bloka u *blockchain*

Hash prethodnog bloka predstavlja rezultat dvostruke primjene SHA-256 hash funkcije, definirane u 1.3, nad zaglavljem prethodnog bloka u lancu. Hash bloka koji je zapravo hash zaglavlja bloka je jedinstveni identifikator svakog pojedinog bloka.

Primijetimo, prema tablici 2.2 i 2.1 hash bloka zapravo nije dio strukture bloka. On se izračunava na strani svakog čvora kada čvor ima potrebe za tim, na primjer kada primi novi blok koji je uključen u lanac. Također, u svrhu vremenske uštede čvor može održavati zasebnu bazu podataka u kojoj su spremljeni hash-evi blokova.

Vremenska oznaka predstavlja vrijeme kada je blok dodan u lanac. Polja težinska oznaka i *nonce* su meta-podaci koji se koriste prilikom dodavanja bloka u lanac i opisani su u poglavlju 3.4. Korijen binarnog hash stabla predstavlja informaciju dobivenu od svih zapisa u bloku.

2.3.2 Binarno hash stablo

Svaki blok u zaglavlju sadrži polje pod nazivom *korijen binarnog hash stabla* koji omogućuje sažet prikaz svih zapisa u bloku ali i jednostavnu provjeru integriteta velikog skupa podataka.

Definicija 2.3.1. *(Binarno stablo)*
Binarno stablo *je konačan skup podataka istog tipa koje zovemo* ***čvorovi****. Pri tome vrijedi:*

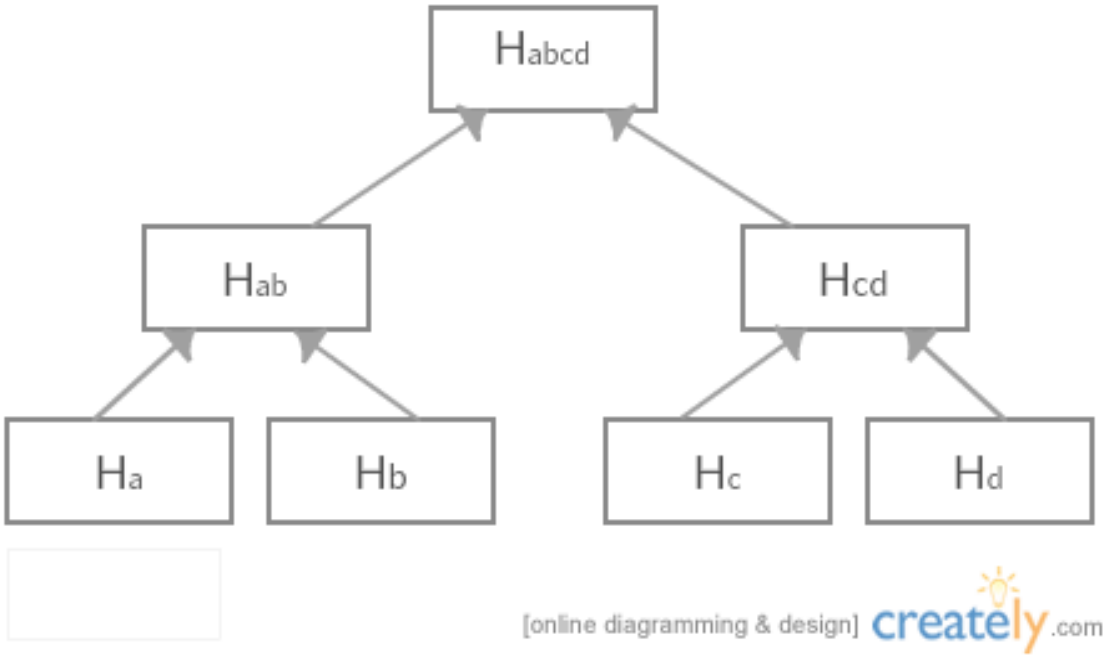
- T *je prazan skup (prazno stablo), ili*

- *postoji istaknuti čvor* r *koji zovemo* ***korijen*** *od* T*, a ostali čvorovi grade uređeni par* (T_L, T_R) *disjunktnih binarnih stabala*

Binarno hash stablo je binarno stablo u kojem su **listovi**, odnosno čvorovi bez djece, hash vrijednosti nekih podataka, a ostali čvorovi su nastali konkatenacijom hash-eva svoje djece te primjenom iste hash funkcije na taj podatak.

U `Bitcoin sustavu` listovi binarnog hash stabla su hash-evi svake pojedine transakcije u bloku. Kao hash funkcija koristi se SHA-256 primijenjena dva puta.

Pohranjivanjem korijena binarnog hash stabla u zaglavlje bloka dobivamo sažeti prikaz svih zapisa u bloku. Nadalje, poznavajući taj sažeti prikaz lako je odrediti pripada li neki zapis bloku bez da imamo uvid u cijeli skup zapisa.



Slika 2.1: Binarno hash stablo

Slika 2.1 prikazuje izgradnju hash stabla u slučaju da bi u bloku bile zapisane četiri transakcije, nazovimo ih *a*, *b*, *c* i *d*. Prema strukturi binarnog hash stabla logično je da izgradnja tog stabla kreće od dna prema vrhu. `Bitcoin sustav`, kao i svaki drugi sustav koji koristi *blockchain tehnologiju*, ima strogo definiran format

podataka koji se upisuju u *blockchain* u ovom slučaju transakcija. No, u hash stablo nisu zapisani ti podaci već hash-evi tih podataka. Pa tako H_a dobivamo na sljedeći način:

$$H_a = \text{SHA-256}(\text{SHA-256}(a)).$$

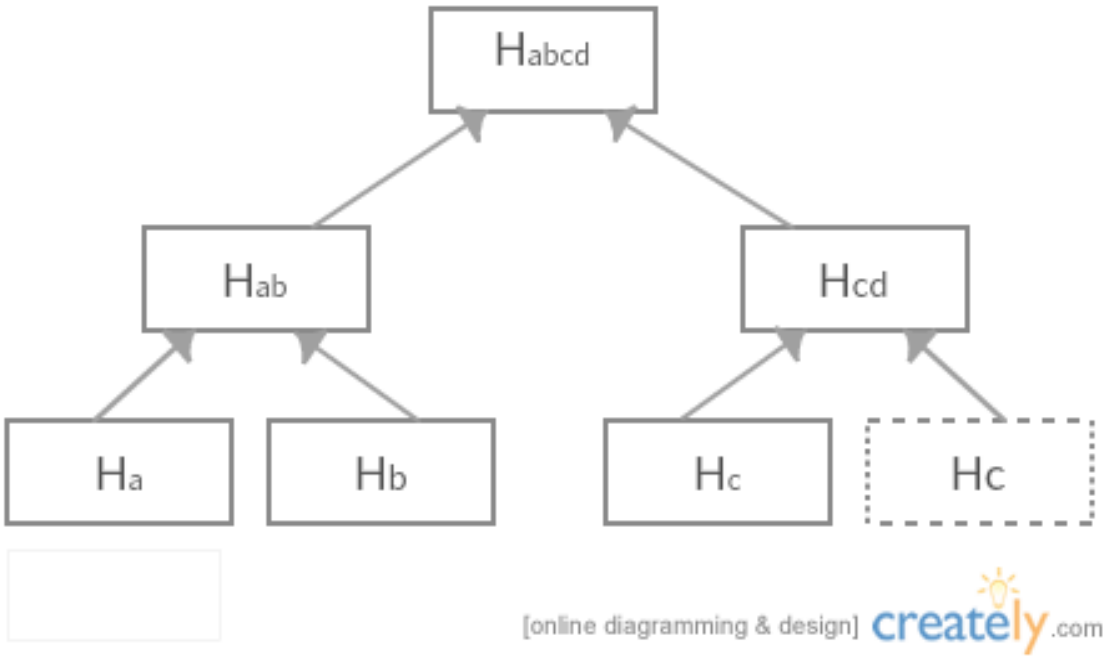
Listove stabla H_b, H_c, i H_d izračunavamo analogno. Sljedeći korak je generiranje čvorova roditelja listova:

$$H_{ab} = \text{SHA-256}(\text{SHA-256}(H_a + H_b)) \text{ i } H_{cd} = \text{SHA-256}(\text{SHA-256}(H_c + H_d))$$

gdje operator + predstavlja konkatenaciju stringova. Nakon toga možemo izračunati hash koji zapisujemo u korijen stabla:

$$H_{abcd} = \text{SHA-256}(\text{SHA-256}(H_{ab} + H_{cd})).$$

Identičan algoritam izgradnje binarnog hash stabla koristi se za bilo koji parni broj transakcija zapisanih u bloku. Ako je u bloku zapisan neparan broj transakcija tada jednostavno dupliciramo zadnju transakciju kao što slika 2.2 prikazuje. Također, napomenimo da je veličina podataka zapisanih u korijenu stabla uvijek 32 bajta bez obzira na broj čvorova.



Slika 2.2: Dupliciranje transakcije u binarno hash stablo

Kasnije ćemo vidjeti da postoje čvorovi u sustavu koji lokalno ne pohranjuju čitav *blockchain*, već samo zaglavlja svih blokova. Hashiranje i primjena binarnih hash stabala ima dvojaku ulogu. Omogućuje provjeru pripadnosti zapisa bloku i lako identificiranje promjene nad podacima.

Čvor koji nema pristup cijelom *blockchainu* može odrediti sadrži li određeni blok neki zapis poznavajući samo taj zapis, odnosno njegov hash i korijen binarnog hash stabla. To određivanje se provodi pomoću *autentikacijskog put* u binarnom hash stablu. Za svaki čvor *binarno stabla* možemo odrediti njegovu razinu. Razina se određuje iz definicije koja kaže da je korijen razine 1, a da su razine djece nekog čvora razine n jednake $n+1$.

Definicija 2.3.2. *(Autentikacijski put)*
Autentikacijski put *je konačan skup* A *čvorova binarnog hash stabla* B *za koje vrijedi:*

- *za svaku razinu stabla* B*, osim za prvu, postoji točno jedan čvor* a *koji pripada skupu* A
- *uz poznavanje još jednog lista stabla* B *koji ne pripada skupu* A *moguće je odrediti korijen binarnog hash stabla* B

Binarna hash stabla spomenutim čvorovima omogućuju provjeru integriteta podataka u bloku. Pogledajmo hash stablo sa slike 2.1 i zamislimo da napadač na sustav iz nekog razloga želi promijeniti transakciju *b*. To bi utjecalo na izgled hasheva $\mathtt{H_b}$, $\mathtt{H_{ab}}$ i $\mathtt{H_{abcd}}$. Ostali pouzdani čvorovi koji čuvaju kopiju *blockchaina* ili samo zaglavlja svih blokova mogu lako utvrditi da je došlo do promijene nad podacima u *blockchainu*.

2.3.3 Povezivanje blokova

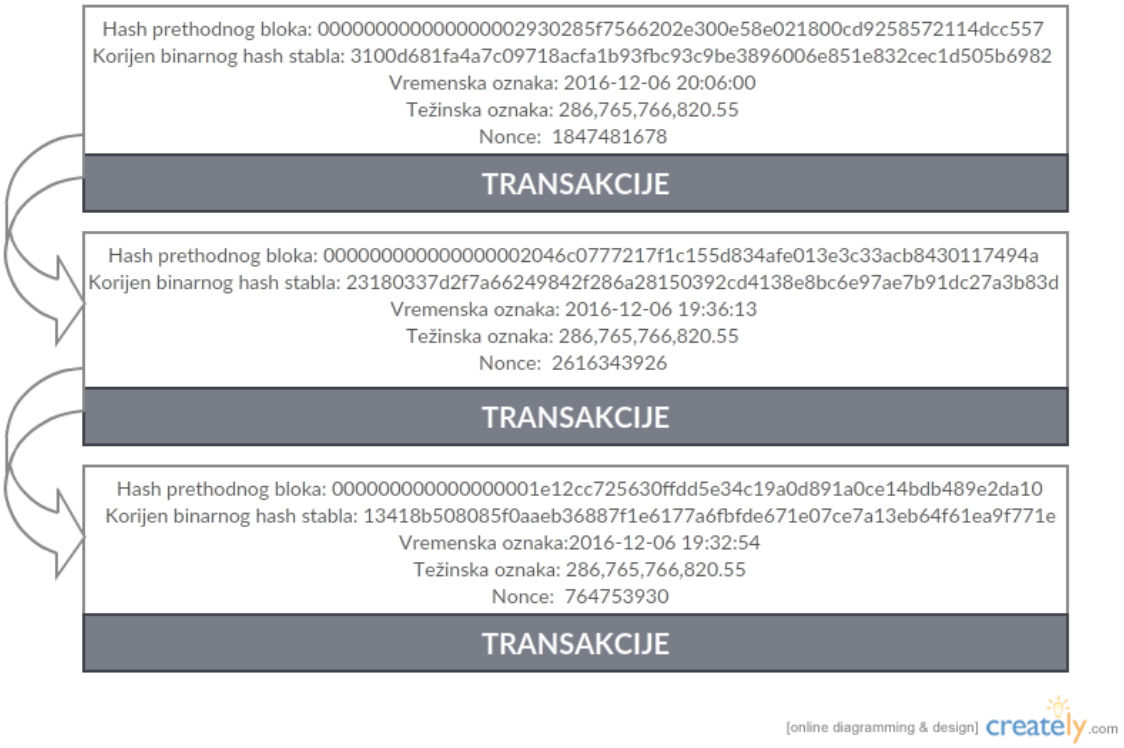
Blokove u lancu možemo usporediti sa stranicama u ovom diplomskom radu. U zaglavlju svake stranice se nalazi ime i broj poglavlja, a na dnu broj stranice. Takvi podaci iz kojih čitamo dodatne informacije zovu se meta-podaci. Zaglavlje jednog bloka u *blockchain* strukturi sadrži tehničke informacije o bloku, referencu na prethodni blok i hash svih podataka sadržanih u bloku dobiven uporabom binarnog hash stabla. Kao i kod diplomskog rada tako i kod *blockchaina* ti podaci su važni za uređivanje i organiziranost.

Ukoliko bi netko istrgnuo sve stranice diplomskog rada lako bismo pomoću brojeva stranica zaključili u kojem ih redoslijedu trebamo čitati, isto tako možemo odrediti razmještaj blokova u lancu pomoću referenci na prethodni blok.

Uporabom hash funkcija umjesto vremenske oznake ili numeriranja dobivamo i validaciju podataka. Svatko tko ima pristup podacima pojedinog bloka ili samo zaglavlju tog bloka može pomoću određene kriptografske funkcije odrediti hash tog bloka. Tako, na primjer, možemo dva puta primijeniti hash funkciju SHA-256 na podatke zaglavlja zadnjeg donjeg bloka sa slike 2.3 i dobivamo sljedeći hash:

`000000000000000002046c0777217f1c155d834afe013e3c33acb8430117494a.`

Primijetimo kako blok koji je kasnije nastao pohranjuje hash bloka koji je neposredno prije njega uključen u lanac. Odgovara li hash zapisan u *blockchainu* hash-u dobivenom na prije spomenuti način tada možemo biti sigurni da su podaci u bloku konzistentni. Ako netko pokuša napraviti izmjenu u podacima mora također mijenjati i sve hash-eve od tog trenutka nadalje. Time bi cijeli *blockchain* izgledao potpuno drugačije.



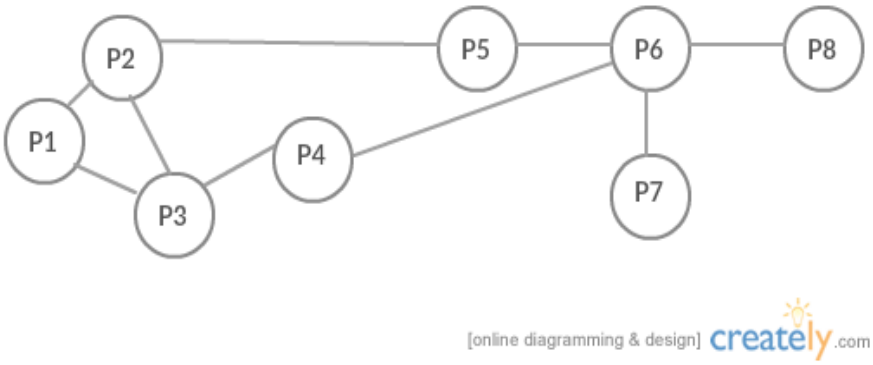
Slika 2.3: Povezivanje blokova

Slika 2.3 prikazuje 3 bloka u `Bitcoin` *blockchainu* pod brojem 442226, 442227, 442228. Svi su nastali u `Bitcoin` verziji 2 što nam nije od velike važnosti za temu ovog rada pa nije navedeno na slici. Ostala polja prate strukturu danu u tablici 2.2.

2.4 Decentralizirani sustav ravnopravnih partnera

Sustav ravnopravnih partnera, odnosno sustav građen prema modelu ravnopravnih partnera (eng. peer-to-peer) sastoji se od velikog broja istovrsnih procesa, takozvanih partnera (eng. peer). Partneri obavljaju zadaće prema potrebama svojih korisnika. Ako je partneru pri obavljanju neke zadaće potrebna pomoć on stupa u komunikaciju sa svojim susjedima, a ti susjedi sa svojim susjedima i tako se komunikacija odvija na razini cijelog sustava.

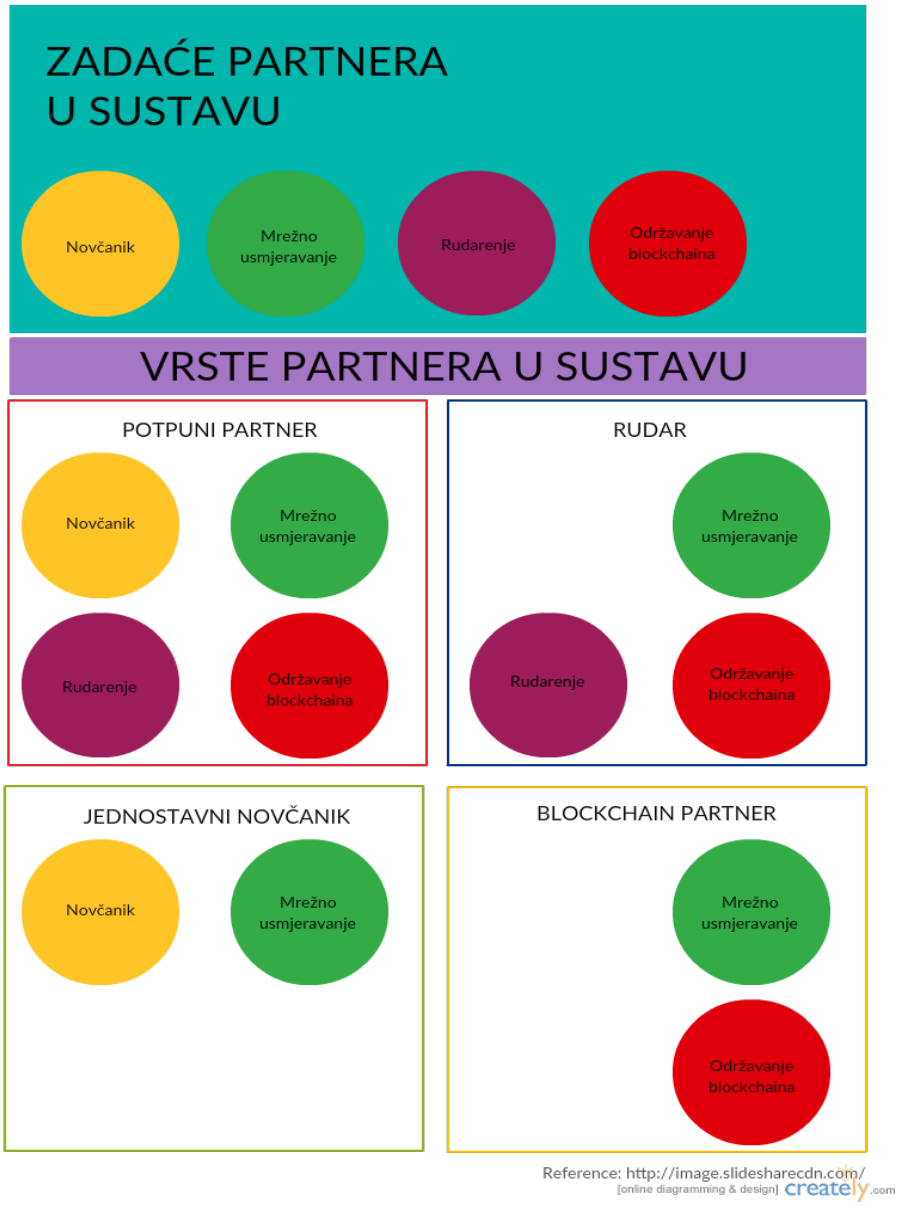
Takvi sustavi mogu se prema strukturi podijeliti na centralizirane i decentralizirane. Centralizirani sustavi su oni kod kojih postoji poslužitelj. Uloga poslužitelja je povezivanje klijenata kako bi oni mogli nastaviti međusobno komunicirati. Glavna karakteristika decentraliziranih sustava je u tome što nema istaknutog poslužitelja. Arhitekturu decentraliziranog sustava možemo vidjeti na slici 2.4.



Slika 2.4: Arhitektura sustava građenog prema modelu ravnopravnih partnera

Sustavi koji koriste *blockchain* tehnologiju spadaju u decentralizirane sustave ravnopravnih partnera. Time je omogućena razmjenu podataka kroz računalnu mrežu pri kojem čvorovi preuzimaju informacije jedni od drugih umjesto s jednog centralnog poslužitelja.

Općenito, sustav ravnopravnih partnera pruža najbolji i najjeftiniji način da veliki broj korisnika dođe do neke datoteke, a troškovi takve komunikacije postaju relativno mali i dijele se među korisnicima.



Slika 2.5: Zadaće i vrste partnera u sustavu

U sklopu *blockchain* tehnologije jedan partner u principu radi na računalu jednog korisnika. Kao što slika 2.5 prikazuje, postoje četiri zadaće koje partner može obavljati. To su: novčanik (wallet), mrežno usmjeravanje (network routing), rudarenje (mining), održavanje cijelog *blockchaina*. U privatnom *blockchain* sustavu u pravilu svaki partner obavlja sve četiri zadaće dok u javnim *blockchain* sustavima

razlikujemo partnere prema zadaćama koje obavljaju. To su sljedeće vrste partnera: potpuni partner, rudar, jednostavni novčanik, blockchain partner.

Kao što je vidljivo sa slike 2.5 sve vrste partnera obavljaju zadaću mrežnog usmjeravanja. Razlog tome je potreba svakog partnera za uspostavljanjem i održavanjem veza s nekim od ostalih partnera u modelu ravnopravnih partnera. Isto tako svaki od partnera koji sudjeluje u sustavu zadužen je za validiranje i difuziju (broadcast) novih zapisa i novih blokova.

U sljedećim potpoglavljima bit će objašnjene uloge blockchain partnera, jednostavnog novčanika i rudara. Potpuni partner može obavljati sve uloge ostalih partnera.

2.4.1 Blockchain partner

Blockchain partner održava *blockchain* sa svim zapisima, počevši od prvog bloka koji se naziva generički blok na koji se nadovezuju svi ostali blokovi sve do zadnjeg kreiranog.

Za razliku od jednostavnog novčanika blockchain partner nema potrebe za oslanjanjem na ostale partnere u svrhu pretraživanja *blockchaina* ili provjere integriteta podataka. Ako je riječ o zapisu transakcija u *blockchain*, blockchain partner u svrhu validacije nove transakcije ima mogućnost provjeriti pripadaju li sredstva koja korisnik želi potrošiti u novoj transakciji zaista tom korisniku. To će napraviti na način da poveže novu transakciju sa svim prijašnjim transakcijama tog korisnika sve do generičkog bloka.

Takav partner oslanja se na ostatak mreže samo kako bi u realnom vremenu primio novokreirane blokove koje nakon toga verificira i nadovezuje na svoju lokalnu kopiju *blockchaina*.

2.4.2 Jednostavni novčanik

U javnim sustavima koji koriste *blockchain* zbog velike količine podataka nema svaki korisnik mogućnost pohraniti čitav *blockchain*. Takav korisnik tada u sustavu sudjeluje kao jednostavan novčanik i na slici 2.5 prepoznajemo ga po tome što među svojim zadaćama nema crveni krug pod nazivom održavanje blockchaina. Glavna zadaća koju jednostavni novčanik obavlja je kreiranje novih zapisa u skladu s protokolom koji propisuje sustav. U svrhu potvrde vlasništva nad digitalnim novce ili nekom drugom digitalnom informacijom čiji integritet želimo zaštiti pohranjivanjem na *blockchain* novčanici pohranjuju parove javnih i privatnih kriptografskih ključeva. Iz javnog ključa se generira adresa koja služi za primanje novca od ostalih korisnika, analogno broju bankovnog računa. Privatne ključevi su potrebni za pristup adresi i novčanim sredstvima. Njih možemo usporediti s osobnim identifikacijskim

brojem (pinom) bankovnog računa i za razliku od adresa nije ih preporučljivo dijeliti s ostatkom sustava.

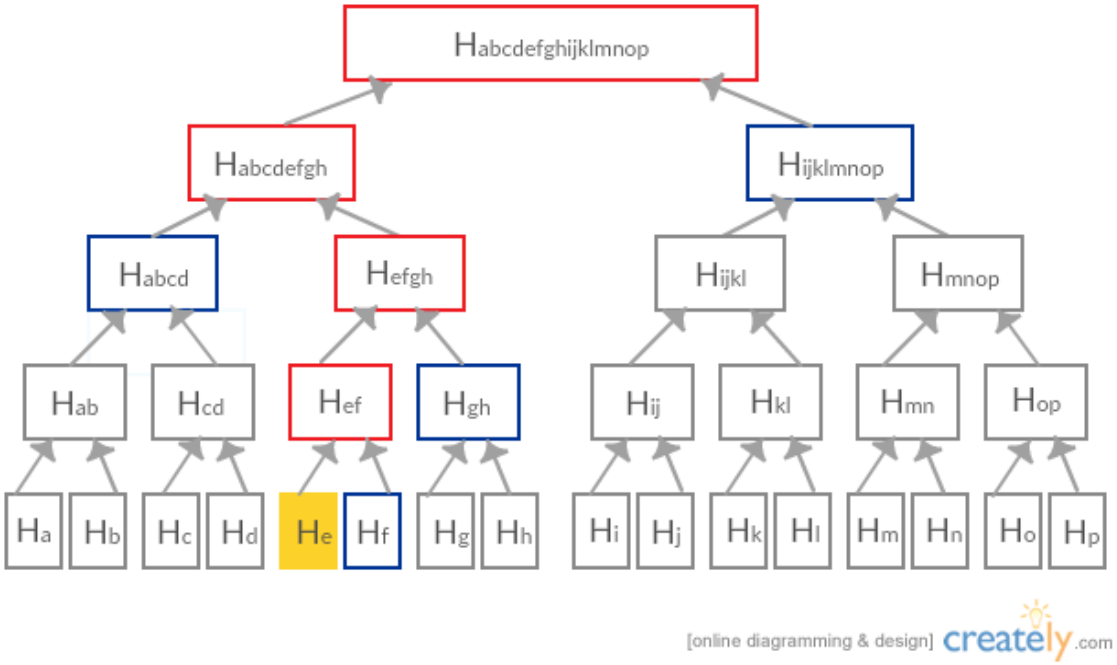
Prije smo napomenuli da svi partneri u sustavu obavljaju validaciju i difuziju novih transakcija i blokova. Kako jednostavni novčanik sam ne sadrži zapis o svim prethodnim transakcijama on provodi takozvanu pojednostavljenu metodu verifikacije.

Pojednostavljena metoda verifikacije podrazumijeva pohranjivanje samo zaglavlja blokova umjesto cijelog *blockchaina* sa svim zapisima. Budući da nemaju punu sliku transakcija koje su se dogodile prije one koju žele validirati oslanjaju se na ostale partnere koji im na njihov zahtjev mogu pružiti uvid u dio *blockchaina*. Jednostavni novčanici provjeravaju dubinu transakcija. Blockchain partner koji konstruira lanac prijašnjih transakcija, vraća se vremenski u prošlost što prema strukturi i načinu povezivanja lanaca u blok podrazumijeva da ide u visinu.

Prema pojednostavljenoj metodi verifikacije jednostavni novčanici verificiraju lanac svih blokova bez transakcija, što su u mogućnosti budući da imaju lokalno pohranjena zaglavlja blokova. Nakon toga verificirani lanac povezuje s transakcijama. Kako bi se uspostavila veza između transakcije i bloka u kojem je ta transakcija zapisana koristi se u put u binarnom hash stablu. Kada jednostavni novčanik otkrije u kojem se bloku nalazi transakcija koju želi validirati on čeka da rudari obave svoj posao i dodaju još šest blokova u *blockchain* kako bi bio siguran da je ostatak mreže potvrdio da se ne radi o nesigurnoj transakciji. Razlog zbog kojeg je potrebno čekati upravo generiranje šest novih blokova vidjet ćemo kasnije.

Binarno hash stablo pruža partnerima novčanicima provjeru kojem bloku pripada određena transakcija, to je prikazano na slici 2.6. U blok, čije binarno hash stablo vidimo na slici, zapisano je 16 transakcija i jednostavni novčanik želi provjeriti pripala li tom bloku transakcija *e*. Budući da poznaje sadržaj transakcije *e* partner lako može odrediti pripadni hash $\mathtt{H_e}$.

Kako bi provjerio pripadnost transakcije bloku potrebno je da primi od nekog od svojih susjednih partnera koji održavaju cijelu kopiju *blockchaina* hash-eve $\mathtt{H_f}$, $\mathtt{H_{gh}}$, $\mathtt{H_{abcd}}$ i $\mathtt{H_{ijklmnop}}$. Ti su hash-evi na slici 2.6 označeni plavim pravokutnicima i nazivaju se *autentikacijski put*. Nakon što jednostavni novčanik posjeduje *autentikacijski put* on može izračunati hash-eve $\mathtt{H_{ef}}$, $\mathtt{H_{efgh}}$, $\mathtt{H_{abcdefgh}}$ i konačno korijen binarnog hash stabla $\mathtt{H_{abcdefghijklmnop}}$ označene crvenim pravokutnicima na način na koji je opisano u 2.3.2. Izračunati korijen stabla partner tada uspoređuje s korijenom binarnog hash stabla zapisanog u zaglavlju bloka, ako se ti podaci podudaraju transakcija *e* pripada bloku sa slike 2.6.



Slika 2.6: Put u binarnom stablu traženja u svrhu potvrde pripadnosti transakcije bloku

Primijetimo da je u ovom slučaju jednostavni novčanik trebao izračunati samo 4 hash-a za provjeru pripadnosti transakcije bloku u kojem je zapisano 16 transakcija. Općenitu, ako je N broj transakcija zapisanih u blok tada je potrebno $\log_2 N$ hash-eva od kojih je svaki veličine 32 bajta. U `Bitcoin` *blockchainu* svaki blok sadrži nekoliko tisuća transakcija. Uzmimo za primjer neki blok koji sadrži 2048 transakcija, veličina tog bloka tada je otprilike 512 kilobajta. Provjera pripadnosti transakcije tom bloku tada iziskuje $\log_2 2048 = 11$ hash-eva veličine 32 bajta, dakle ukupno 352 bajta. Iz ovog primjera vidimo da binarna hash stabla omogućuju veliku prostornu uštedu uz mogućnost provjere integriteta podataka.

2.4.3 Rudar

Partneri rudari preuzimaju nove zapise koje su kreirali novčanici, formiraju ih u blokove i dodaju u *blockchain*. U `Bitcoin` protokolu dodavanje novih zapisa iziskuje korištenje računalnih resursa. Kada rudar doda blok u *blockchain*, rješavajući algoritam pod nazivom proof-of-work (vidi 3.4) i koristeći svoje računalne resurse, tada sve transakcije u tom bloku postaju potvrđene, a rudar za nagradu dobiva određeni broj novih bitcoina.

Poglavlje 3

Algoritmi za postizanje konsenzusa

3.1 Uvod

Pojam konsenzus predstavlja dogovor procesa u sustavu oko vrijednosti jedne varijable. U distribuiranim sustavima bez centralnog autoriteta kakve smo mi do sada promatrali nameće se izazov kako postići da se svi partneri slože oko stanja *blockchaina* bez da vjeruju jedni drugima. Postizanje tog konsenzusa odnosno dogovora jest najvažnija stavka sigurnosti *blockchaina*. Algoritmi za postizanje konsenzusa daju rješenje za taj izazov. Usuglašavanje oko stanja *blockchaina* spada u kategoriju problema koji je u računarstvu poznat pod nazivom *problem usuglašavanja bizantskih generala*. U ovom poglavlju opisan je taj problem i najpoznatiji algoritam za postizanje konsenzusa u sinkronoj mreži. Također su opisana dva algoritma pomoću kojih sustavi koji koriste *blockchain* mogu riješiti navedeni problem.

3.2 Problem usuglašavanja bizantinskih generala

U distribuiranim sustavima može doći do različitih grešaka u radu procesa. Te greške možemo podijeliti u 3 skupine, to su:

- prestanak rada procesa,
- propust u radu procesa,
- Bizantska greška.

Bizantska greška je najteži oblik greške. Proces griješi tako što se ponaša sasvim nepredvidljivo. On može stati s radom, pa opet krenuti, propuštati slanje ili primanje poruka, krivo računati, slati korumpirane poruke.

Najpoznatiji problem postizanja konsenzusa u distribuiranom sustavu uz prisustvo bizantinskih grešaka jest *problem usuglašavanja bizantskih generala.*

Slijedi neformalni opis tog problema. N vojnih jedinica okružuje jedan dvorac i ima ga namjeru napasti. Svaku vojnu jedinicu predvodi po jedan general i postoji jedan vodeći general odnosno kraj. Unutar dvorca nalazi se neprijateljska vojska koja brani dvorac. Cilj je postizanje dogovora između N generala o vremenu napada na dvorac. Budući da se ovaj scenarij odvija u dobu kada nema mobitela i mogućnosti da generali nazovu jedni druge, oni dogovaraju vrijeme napada slanjem poruka koje prenosi glasnik na konju. Neki od generala su izdajice i njihova vojna jedinica bori se na strani neprijateljske vojske. Postoji najviše f od N generala koji su izdajnici. Ostali generali ne znaju koji su generali izdajnici i nema načina da to saznaju. Vojne jedinice lojalnih generala dovoljno su jake da preuzmu dvorac, ali uz uvjet da su njihove akcije koordinirane, sve jedinice u isto vrijeme kreću u napad.

Neka se, na primjer, unutar dvorca nalazi 300 vojnika i neka dvorac okružuje 5 vojnih jedinica s po 100 vojnika. Jedino general koji vodi 3. vojnu jedinicu je izdajnik, nazovimo ga G3. Prema tome, generali G1, G2, G4 i G5, odnosno vođe 1., 2., 4. i 5. vojne jedinice lojalni su kraju. Kada bi svi lojalni generali napali dvorac u isto vrijeme njihovih 400 vojnika imalo bi šanse poraziti vojsku od 300 vojnika koja brani dvorac, čak i u slučaju da se braniteljskoj vojsci pridruži 100 vojnika generala G3.

Cilj generala G3 je izbjeći jednako vrijeme napada svih lojalnih vojnih jedinica i to može lako postići. Promotrimo sljedeći niz događaja:

1. G1 šalje poruku *"Napad u 16 sati."* prema G2,
2. G2 šalje tu istu poruku *"Napad u 16 sati."* prema G3,
3. G3 je izdajica, on mijenja sadržaj poruke u *"Napad u 15 sati."* i šalje je prema G4,
4. G4 šalje poruku *"Napad u 15 sati."* prema G5.

Nakon razmjene poruka generali G4 i G5 napadaju dvorac s 200 vojnika u 15 sati, nadajući se da će im se ostatak vojske pridružiti. Budući da su generali G1 i G2 uvjereni da će se napad održati u 16 sati njihove vojne jedinice ne priključuju se napadu u 15 sati. Vojna jedinica generala izdajnika se može u bilo kojem trenutku priključiti vojsci branitelja, tada ukupno 400 vojnika brani dvorac. 200 vojnika generala G4 i G5 u 15 sati je nema šanse protiv vojnika koji brane dvorac i oni gube bitku. Također u 16 sati 200 vojnika iz 1. i 2. vojne jedinice gubi bitku od brojčano moćnije vojske branitelja.

Promatramo li sustav koji koristi *blockchain tehnologiju* u terminima prije opisanog problema. Partneri u distribuiranom sustavu predstavljaju generale. Digitalni podaci koje želimo upisati u *blockchain* su poruke među generalima. *Blockchain* ima ulogu pohrane dogovorenog vremena napada. Pojedini partner ne zna broj ostalih partnera kao ni koji su od partnera izdajnici. Izdajnicima je u interesu upisati u *blockchain* podatke koji nisu istiniti. Algoritmi za postizanje konsenzusa omogućuju partnerima da u ovakvim uvjetima budu sigurni da su podaci koji se upisuju u nove blokove točni te da su podaci prije zapisani u *blockchain* istiniti i nepromijenjeni.

3.3 Lamportov algoritam

Lamportov algoritam je najpoznatiji algoritam za postizanje konsenzusa u sinkronoj mreži. U ovom slučaju radi jednostavnosti promatramo situaciju kada se generali dogovaraju oko odluke napad ili povlačenje. Tako je potrebno postići konsenzus oko varijable veličine jednog bita. Napomenimo, da se generali dogovaraju o vremenu napada koje se može prikazati varijablom tipa `integer`, tada bi bilo potrebno postići dogovor oko svakog od 32 bita te varijable.

Problem usuglašavanja bizantinskih generala zahtijeva da oblikujemo algoritam pomoću kojeg svi lojalni generali odabiru istu akciju. Generale poistovjećujemo s partnerima ili čvorovima u sustavu. Lamportov algoritam se sastoji od *f+1* koraka od kojih se svaki korak dijeli na tri faze. Prisjetimo se, f je broj generala izdajnika dok je N ukupan broj generala. Ovaj algoritam zahtijeva da je $N > 4f$. Svaki partner ima svoj prijedlog odluke (0 ili 1) koji se može promijeniti u svakom koraku, a inicijaliziran je nekom ulazom vrijednošću. Na kraju algoritma, svi lojalni partneri imat će usuglašene prijedloge odluke.

Algoritam je zasnovan na rotirajućem koordinatoru kralju. Tijekom koraka k ulogu kralja igra partner P_k. Faze svakog koraka su sljedeće:

1. Svaki partner šalje svoj prijedlog odluke svim drugim partnerima. Partner svoj prijedlog i primljene prijedloge drugih partnera zapisuje u lokalno polje `V[ ]`, tako da `V[j]` sadrži prijedlog od P_j.

2. kralj oblikuje svoj kraljevski prijedlog, tako da pronađe većinsku vrijednost u lokalnom polju `V[ ]`, ili tako da uvaži defaultnu ako nema većine. Kralj šalje svoj prijedlog svim drugim partnerira. Partneri čuvaju kraljev prijedlog u lokalnim varijablama *kingValue*.

3. Svaki partner oblikuje svoj novi prijedlog odluke i sprema ga u lokalnu varijablu *myValue*. Novi prijedlog odluke je ili većinska vrijednost iz lokalnog `V[ ]` ili kraljev prijedlog iz *kingValue*. Partner se odlučuje za lokalnu vrijednost iz `V[`

`]` ako se ona pojavljuje u `V[ ]` na više od $N/2 + f$ mjesta, inače se odlučuje za *kingValue*. Na kraju treće faze partner izračunati *myValue* upisuje na odgovarajuće mjesto u lokalno polje
`V[ ]`, tako da bi poslužio kao inicijalni prijedlog odluke u idućem koraku.

U [11] možete pronaći dokaz korektnosti algoritma u slučaju $N > 4f$. Primijetimo da ovim algoritmom možemo postići dogovor između generala G1, G2, G3, G4 i G5 iz prethodnog primjera. Ali samo zato što nam je poznato da je jedan general izdajnik, odnosno $f = 1$, dok je $N = 5$ i vrijedi $5 > 4 * 1$.

3.4 Proof-of-work

Proof-of-work algoritam nastao je u sklopu kriptovalute `Bitcoin` i svodi se na rješavanje kriptografskog problema u zaglavlju bloka. Problem rješavaju partneri rudari koristeći računalne resurse u procesu koji se naziva rudarenje (eng. mining) i time uključuju novi blok u *blockchain*.

Proces rudarenja podrazumijeva traženje broja *nonce* pomoću kojeg će rudar, zajedno s ostalim podacima iz bloka, izračunati hash koji zadovoljava određene kriterije. Rudari se međusobno natječu, tko će prvi izračunati takav hash. Onaj rudar kojem to uspije šalje novi blok difuzijom kroz sustav i biva nagrađen za svoj rad. Kriteriji se mogu mijenjati s obzirom na vrijeme koje je bilo potrebno da se prijašnji blokovi uključe u lanac kako bi se postigao jednak razmak između dodavanja novih blokova.

Postavlja se pitanje koja je uloga proof-of-work algoritma i korištenja računalnih resursa pri dodavanju novih blokova. Kada za dodavanje novog bloka u lanac ne bi bilo potrebno provoditi proof-of-work algoritam tada bi najbrže računalo u mreži uvijek moglo dodati novi blok i poslati ga ostalim partnerima. Što bi značilo da cijeli proces više nije decentraliziran i da točnost podataka koje zapisujemo u blok ovisi o jednom ili nekoliko partnera iz mreže. Izračunavanje hash-eva s različitim vrijednostima broja *nonce* služi kao dokaz da rudar aktivno sudjeluje u obradi novih zapisa i pronalaženju rješenja za blok u koji se ti novi zapisi pohranjuju.

`Bitcoin sustav` zahtijeva da izlazni hash proof-of-work algoritma bude manji ili jednak od broja koji se naziva *cilj*. Činjenicu da je hash manji od broja *cilj* možemo interpretirati i time da hash započinje određenim brojem nula. *Cilj* se izračunava iz težinske oznake zapisane u zaglavlje bloka i automatski se prilagođava kako bi se osiguralo da vrijeme potrebno za prihvaćanje bloka bude otprilike deset minuta. Trenutni *cilj* se dobije kao kvocijent *početnog ili baznog cilja* koji iznosi
`00000000FFFF0000000000000000000000000000000000000000000000000000` i težine zapisane u zaglavlju bloka.

Proof-of-work algoritam jednostavno možemo opisati sljedećim koracima:

1. Skupi nove transakcije i kreiraj novi blok.
2. Odaberi broj *nonce*.
3. Izračunaj hash zaglavlja novog bloka i broja *nonce*.
4. Provjeri je li hash iz koraka (3.) manji od trenutnog broja *cilj*. Ako da, pošalji novi blok ostalim partnerima u sustavu. Ako ne, vrati se na korak (2.).

3.4.1 Hash funkcija

Hash funkcije kao argument uzimaju podatke varijabilne duljine, dok je rezultat uvijek fiksne duljine. *Blockchain* tehnologija iskorištava važna svojstva hash-eva, odnosno vrijednosti hash funkcija. Vrlo je lako proizvesti hash iz podataka kao što je `Bitcoin` blok, ali je gotovo nemoguće otkriti koji su to podatci gledajući samo hash. Vrlo lako je proizvesti hash iz velike količine podataka, ali svaki hash je jedinstven. Ako se promijeni samo jedno slovo ili brojka, hash se kompletno mijenja. Uzmimo tekst *Ovo je diplomski rad o blockchain tehnologiji*, na njega nadodajemo broj X gdje je *X=0, 1, 2, ...* i računamo vrijednost SHA-256 hash funkcije.

```
SHA-256("Ovo je diplomski rad o blockchain tehnologiji0") =
    4a8140c36617e405529468263582cd553b8ba949fff2a4ee4466eb769d054e21

SHA-256("Ovo je diplomski rad o blockchain tehnologiji1") =
    a44d0a6b0c79f53827f9a99d1bc8d36474106f638d411cf4cf629d3e0754d94e

SHA-256("Ovo je diplomski rad o blockchain tehnologiji2") =
    3758b36cd3f7113090decd8267914bf2a45166537601130a081d190e65f31717

SHA-256("Ovo je diplomski rad o blockchain tehnologiji3") =
    16682e08d6c5fdc590acf40c49a9c24c351465d576c50951b69abe75038f96c5

SHA-256("Ovo je diplomski rad o blockchain tehnologiji4") =
    ba891692b7642d087deF7f7267c9732487f73d7269e0caa6b58ed54496f06565d

SHA-256("Ovo je diplomski rad o blockchain tehnologiji5") =
    b3e213fbc4b84d315c4e16fb64695c8dc087ffd1cfdeb522f73d4f56b7d31e17

...
```

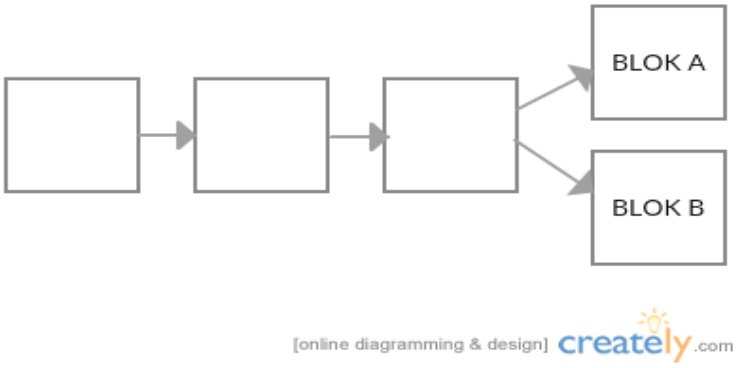
```
SHA-256("Ovo je diplomski rad o blockchain tehnologiji20") =
    9c3b4fa42488da75f47849c455e0bd7ca4d6416c65a4bb1e16d54dcfb3edcdce

SHA-256("Ovo je diplomski rad o blockchain tehnologiji21") =
    098bd24561e3491df483fd8b41db24b2362588ae558d1f943aaec8edaf11c21a
```

Želimo li u navedenom primjeru pronaći hash koji započinje znamenkom 0 ili koji je u numeričkom smislu manji od heksadecimalnog broja 0x1000000000000000000000000000000000000000000000000000000000000000 potrebno je doći do vrijednosti X=21. Da je traženi broj 0 na početku hash-a bio veći trebalo bi nam dulje vrijeme da pronađemo X uz koji bi izračunati hash bio zadovoljavajući.

Ako pogledamo korake algoritma koje smo prije naveli, posao partnera rudara je vrlo sličan ovom iz prethodnog primjera. Oni najprije prikupljaju nove zapise koje treba upisati u *blockchain* i konstruiraju novi blok. Na podatke iz zaglavlja tog bloka, koje smo opisali u tablici 2.2 nadodaju redom različite vrijednosti broja *nonce* te na taj podatak dva puta primjenjuju SHA-256 funkciju sve dok rezultantni hash ne postane manji od broja *cilj*. Rudar koji prvi pronađe zadovoljavajući hash to javlja ostalim partnerima u sustavu, kako bi oni prestali raditi na uključenju tog bloka u *blockchain*.

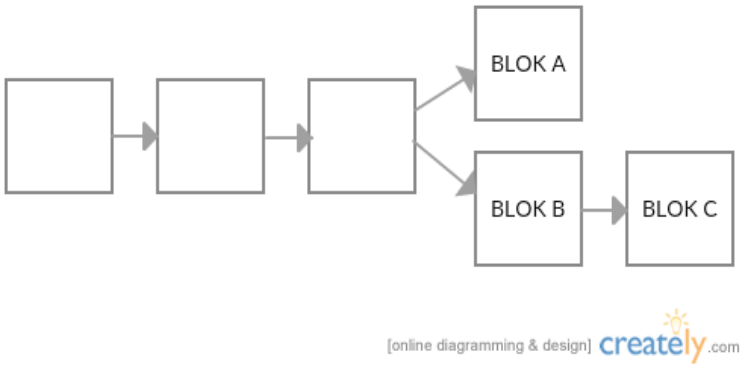
3.4.2 Račvanje u blockchainu

Zamislimo situaciju kada dva rudara, nazovimo ih R1 i R2 u relativno kratkom vremenskom razdoblju izračunaju zadovoljavajući hash. Neka je R1 izračunao zadovoljavajući hash za blok koji ćemo nazvati *BlokA*, a R2 za *BlokB*. Relativno kratko vremensko razdoblje znači da rudar koji je prvi našao rješenje proof-of-work algoritma nije stigao obavijestiti ostale partnere u sustavu da je ova runda dodavanja bloka gotova i da drugi rudari mogu početi raditi na dodavanju novog bloka. Rudar R1 tada šalje *BlokA* kroz mrežu, dok s druge strane rudar R2 šalje *BlokB*. Neki partneri će najprije dobiti novi blok rudara R1 tj. *BlokA* i dodati ga u svoju lokalnu kopiju *blockchaina*, dok će drugi dodati *BlokB* za koji je rješenje proof-of-work algoritma našao rudar R2. Nakon nekog vremena partneri u sustavu će primiti i blok drugog rudara, tada dolazi do račvanja u *blockchainu*.



Slika 3.1: Račvanje u blockchainu

U takvoj situaciji svi partneri u sustavu prate oba lanca, budući da su oba bloka dobiveni rješenjem proof-of-work algoritma i oba su pravilni blokovi. Ti blokovi u lancu tada imaju istog roditelja, kao što je prikazano na slici 3.1 ali rudari rade na onoj grani *blockchaina* na kojoj se nalazi blok koji su prvi prihvatili. Tako otprilike pola partnera rudara pokušava dodati novi blok na *BlokA*, a druga polovica na *BlokB*. Pretpostavimo da je rudar R3 uspio dodati blok pod nazivom *BlokC* na *BlokB* kao na slici 3.2.



Slika 3.2: Dodavanje bloka nakon račvanja

Svi partneri u sustavu pridržavaju se takozvanog *pravila najduljeg lanca*. Duljina lanca ovisi o tome koliko je rada na proof-of-work algoritmu potrošeno na kreiranje

lanca, što se dobije sumom težinskih oznaka blokova koji čine taj lanac. Prisjetimo se težinske oznake blokova zapisane su u zaglavlju bloka pa svaki partner može izračunati duljine lanaca. Nakon što partneri prime vijest o tome da je rudar R3 dodao novi blok tada donji lanac sa slike 3.2 odnosno lanac koji sadrži *BlokB* i *BlokC* postaje najdulji. Rudari koji su do sad radili na produljenju tog lanca to nastavljaju činiti i dalje. Ostatak rudara zbog *pravila najduljeg lanca* mora odbaciti *BlokA* i nastaviti na izgradnji istog linearnog lanca. Zapisi na Bloku A se ignoriraju i ne smatraju se dijelom *blockchaina*.

U `Bitcoin sustavu` zbog račvanja i činjenice da neke transakcije mogu biti zapisane na kraćem lancu i time odbačene nakon nekog vremena postoji pravilo da je transakcija valjana ako su zadovoljena sljedeća dva uvjeta:

- transakcija je zapisana na najdužem lancu u *blockchainu*,
- nakon bloka u kojem je zapisana transakcija dodano je još 5 blokova u *blockchain*.

Moguće je da su transakcije koje su bile zapisane u bloku koji je odbačen već upisane u *blockchain* u blok koji je nastao u otprilike isto vrijeme kao i odbačeni. Ako to nije slučaj bit će zapisane nakon nekog vremena jer su partneri rudari primili tu transakciju i ona se nalazi među kandidatima za novi blok.

3.5 Proof-of-stake

Proof-of-stake je još jedan algoritam za postizanje konsenzusa, odnosno dogovora o trenutnom stanju *blockchaina* među partnerima u distribuiranom sustavu. Nastao je od strane kriptovaluta koje su konkurencija `Bitcoinu`, nešto kasnije od proof-of-work algoritma. Mnogi informatički stručnjaci koji se bave proučavanjem kriptovaluta kao najveću zamjerku proof-of-work algoritma navode veliku potrošnju računalne snage i mogućnost rudarenja takozvanih vanjskih partnera. Pod vanjskim partnerom se podrazumijeva partner koji ne sudjeluje u stvaranju zapisa u *blockchain* već samo posjeduje dovoljno dobro računalo da bi mogao biti konkurentan u stvaranju novih blokova. Proof-of-stake algoritam pokušava eliminirati tu slabost sužavanjem kruga partnera koji imaju pravo generiranja novih blokova i manjom težinom algoritma hashiranja koju ti partneri moraju zadovoljiti kod generiranja novog bloka.

Postoji više načina na koji se određuje tko ima pravo uključivati novi blok u *blockchain*. Ovdje je opisan način koji koristi digitalna kriptovaluta `Peercoin`. Dok proof-of-work podrazumijeva korištenje računalne snage kako bi se dodao novi blok u *blockchain* proof-of-stake algoritam omogućava kreiranje novog bloka onim korisnicima koji posjeduju određenu količinu novca u sustavu. Kao i u slučaju proof-of-work

algoritma validatori se natječu tko će prvi izračunati zadovoljavajući hash. Generiranje novog bloka uključuje slanje `Peercoin`-ova samom sebi, kako bi se dokazalo da validator posjeduje određeni iznos novca.

Najvažnija veličina koja se koristi u `Peercoin sustavu` je *starost novca*. Starost novca jednostavno je definirana kao umnožak količine novca i dana koji taj novac stoji u novčaniku partnera nepotrošen. Ako neki partner u sustavu posjeduje 80 `Peercoin`-ova koje nije potrošio u proteklih 10 dana tada je starost tog novca 800 dana. U trenutku kada partner uplati taj novac nekom drugom partneru u sustavu (ili sam sebi) kažemo da je starost novca uništena odnosno starost je jednaka 0. Kako bi se lakše utvrdila starost novca svaka transakcija u `Peercoin sustavu` mora sadržavati vrijeme izvršenja.

Kod proof-of-stake algoritma partneri koji imaju mogućnost uključivanja novih blokova u *blockchain* nazivaju se još i validatori. Validatori na neki način daju svoj novac kao polog za obavljanje proof-of-stake algoritma. Postoji minimalni iznos jedinica kriptovalute koji čvor treba posjedovati kako bi postao kandidat za kreiranje novog bloka odnosno validator. Taj minimalni iznos se naziva *cilj* i u sustavu se prilagođava tako da vrijeme između dodavanja dva nova bloka bude otprilike jednako. Primijetimo, kod proof-of-work algoritma *cilj* je predstavljao broj u heksadecimalnom zapisu pomoću kojeg se definirala težina izračunavanja zadovoljavajućeg hash-a.

Kod proof-of-stake algoritma broj od kojeg zadovoljavajući hash mora biti manji se mijenja kod svakog novog bloka i za svakog partnera je drugačiji. Glavnu ulogu u definiranju težine pronalaska zadovoljavajućeg hash-a imaju prije spomenuti *cilj* i *starost novca* prema sljedećoj formuli:

```
zadovoljavajući hash < starost novca * cilj.
```

Starost novca iz formule je starost onog novca koji partner koji se natječe u generiranju novog bloka uplaćuje sam sebi kako bi dokazao da je on podoban validator.

Pogledajmo primjer kada se dva validatora V1 i V2 natječu u kreiranju zadovoljavajućeg hash-a. Neka validator V1 u novonastali blok uključuje transakciju kojom sebi uplaćuje 100 Percoin-ova starih 10 dana, a validator V2 kao dokaz da posjeduje dovoljno novaca za rudarenje sebi uplaćuje 900 `Peercoin`-ova starosti 7 dana. Budući da je *cilj* u datom trenutku jednak za sve validatore, V1 mora proizvesti hash manji od broja `1000*cilj`, a V2 hash manji od `6300*cilj`. Naravno, V2 ima puno veće šanse za pobjedu u ovoj rundi dodavanja bloka jer se natjecao s novcem veće starosti. Ako validator V2 zaista pobjedi i njegov blok bude uključen u *blockchain* transakcija kojom sam sebi uplaćuje 900 `Peercoin`-ova također postaje valjana i taj novac sada ima starost 0. Uništavanje starosti novca ima za posljedicu to da validator V2 gotovo pa nema šanse za pobjedu pri uključivanju sljedećih nekoliko blokova. V2 mora pre-

pustiti ostalim partnerima kreiranje novih blokova, dok ne prođe dovoljno vremena da starost njegovog novca ponovo bude konkurentna.

Starost novca u `Peercoin sustavu` ima još jednu ulogu. Ako dođe do račvanja u *blockchain* ukupna starost novca u cijelom lancu određuje na koju granu treba nastaviti dodavati blokove. Kako svaka transakcija u bloku sadrži informaciju o starosti novca, sumiranjem tih vrijednosti lako je izračunati ukupnu starost novca po bloku i u cijelom lancu. Lanac s najvećim ukupnim zbrojem partneri u sustavu izabiru kao glavni lanac.

3.6 Rješenje problema Bizantskih generala

Pretpostavimo da bizantski generali za dogovor oko vremena napada koriste *blockchain* i zapis novih informacija pomoću proof-of-work algoritma. Svaki general kada prvi puta primi poruku s vremenom napada za koje još nije čuo pokrene na svom računalu izračunavanje hash-a koji sadrži informaciju o tom vremenu. Broj *cilj* od kojeg taj hash mora biti manji kako bi postao zadovoljavajuć za upisivanje zapisa u *blockchain* je tako prilagođen da je potrebno 10 minuta da jedan general nađe rješenje.

Kada neki general nađe zadovoljavajući hash pošalje ga difuzijskom porukom ostalima u mreži, kako bi ga oni upisali u lokalnu kopiju *blockchaina*. Ostali generali tada počinju izračunavati novi hash koji u sebi sadrži prethodno izgenerirani hash. Na taj način generali žele izgraditi što dulji lanac s istim vremenom napada i poštuju pravilo da se u jednoj grani lanca nalaze samo hash-evi koji imaju u sebi zapisano jedno vrijeme napada. Ako neki general izdajnik želi upisati neko drugo vrijeme napada on mora potaknuti račvanje u *blockchainu*. No, budući da ima više generala odanih kraju prema *pravilu najduljeg lanca* grana generala izdajnika će nakon nekog vremena biti ignorirana.

Nakon 2 sata, postoji lanac odnosno dio *blockchaina* s 12 hash-eva s istim vremenom napada. Svaki general sada ima dokaz da se na izgradnju tog lanca potrošila određena količina računalne snage i može vjerovati da je vrijeme napada zapisano u tom lancu dogovoreno od strane svih generala.

Kod proof-of-stake algoritma imamo sličnu situaciju. Razlika je u tome da novo vrijeme napada neće prvi objaviti nasumično odabrani general koji je prvi izračunao zadovoljavajući hash već onaj kojem kralj najviše vjeruje ili onaj koji ima vlasništvo nad najviše parcela u zemlji. Takav general ima najveći ugled i hash koji on mora izračunati mora zadovoljiti slabije kriterije od hash-eva koje moraju izračunati ostali generali. Nakon njega ostali generali kreću u izračunavanje novog hash-a koji sadrži isto vrijeme napada kao i prethodni. U tome najveće šanse za novi blok u lancu ima sljedeći general po ugledu.

Poglavlje 4

Slučajevi upotrebe Blockchaina

4.1 Uvod

Iako je *blockchain tehnologija* nastala za potrebe kriptovalute `Bitcoin` danas ona ima široku primjenu. Ovo poglavlje najprije donosi pregled najpoznatijih kriptovaluta koje su se razvile nakon `Bitcoina`. Za svaku takvu kriptovalutu naznačene su njene karakteristike po kojima se razlikuje od ostalih kriptovaluta. U nastavku je opisano što su to pametni ugovori i kako funkcionira Ethereum, najpoznatija platforma za izvršavanje pametnih ugovora.

4.2 Kriptovalute

Kriptovalute (eng. cryptocurrency) su oblik digitalne valute koje funkcioniraju na temelju na kriptografskih algoritama. Glavna karakteristika kriptovaluta je da ne postoji središnja institucija poput banke koja njome upravlja. Već se koristi *blockchain tehnologija* kako bi se postigao potpuno decentralizirani sustav. Nove jedinice kriptovalute proizvode se rudarenjem, kao nagrada za novokreirani blok. Korisnici mogu doći u posjed kriptovalute kupovinom određene svote kriptovalute, prodajom dobara ili usluga. Budući da kriptovalute pružaju relativnu anonimnost svojim korisnicima često ih se povezuje s ilegalnim aktivnostima kao što su trgovanje oružjem, ljudima ili narkoticima.

Do sada smo spomenuli `Bitcoin` i `Peercoin`. `Bitcoin` je prva kriptovaluta u povijesti. Predstavljen je 2008. godine člankom pod imenom "Bitcoin: A Peer-to-Peer Electronic Cash System" [10], pod pseudonimom Satoshi Nakamoto. Kao što smo već spomenuli u prosjeku svakih 10 minuta se generira novi blok na `Bitcoin` *blockchainu*, to je i prosječno vrijeme za potvrdu transakcije. Nakon generiranja novog bloka partner rudar dobiva nagradu koja trenutno iznosi 12.5 BTC-a (BTC

je skraćenica za jedinicu kriptovalute `Bitcoin`) kao i naknade za transakcije koje su zapisane u novi blok uplaćene od strane korisnika. Rudarenje je jedini način na koji se izdaju novi BTC-ovi. Definiran je i maksimalan broj jedinica `Bitcoin` kriptovalute, koji je 21 milijun. U tome vidimo još jednu razliku u odnosu na klasične valute, koje centralne banke mogu izdavati po potrebi i procjeni.

`Peercoin` je nastao 2012. godine i prva je kriptovaluta kod koje se koristi proof-of-stake algoritam za postizanje konsenzusa. Uz proof-of-stake koristi se i proof-of-work algoritam. `Peercoin` nema definiranu gornju granicu za maksimalnu količinu izdane valute, ostale karakteristike su slične `Bitcoinu`.

Budući da je `Bitcoin` algoritam objavljen kao OpenSource kod (svatko ga može preuzeti i modificirati), mnoge nove kriptovalute su stvorene na taj način. Kako bi se razlikovale od `Bitcoina` i poboljšale njegova svojstva, nove valute obično mijenjaju neke od karakteristika `Bitcoin sustava` kao što su: maksimalna količina valute, nagrada po bloku, vrijeme potvrde transakcije, naknada za transakcije itd. Slijedi kratak pregled nekih od ostalih popularnih kriptovaluta.

Litecoin

Nastao je 2011. godine i druga je najpopularnija kriptovaluta nakon `Bitcoina`. Umjesto SHA-256 funkcije u proof-of-work algoritmu `Litecoin` koristi *skripte*. *Skripta* je također kriptografska funkcija, ali znatno jednostavnija od SHA-256. Zato je rudarenje temeljeno na skriptama brže i jednostavnije ali s druge strane takav sustav je više osjetljiv na sigurnosna pitanja. U `Litecoin` sustavu dodavanje novog bloka u *blockchain* odvija se u prosjeku svaki 2,5 minuta, pa se tako proizvodi i više jedinica novca i ukupna definirana količina novca koja će biti izdana je 84 milijuna.

Primecoin

`Primecoin` je nastao 2013. godine i spada u novu generaciju kriptovaluta čiji tvorci smatraju proof-of-work algoritam za postizanje konsenzusa beskorisnim. Takvi sustavi pokušavaju iskoristiti računalnu snagu koja se troši pri kreiranju novog bloka na rješavanje nekog drugog korisnog problema. Uz traženja zadovoljavajućeg hash-a partneri rudari u `Primecoin sustavu` grade lanac prostih brojeva. Takvim načinom rada rudari otkrivaju i izračunavaju sve veće i veće proste brojeve koji se mogu koristiti u drugim znanstvenim disciplinama. Svaki blok u `Primecoin` *blockchainu* ima zapisan i otkriveni prosti broj, tako se paralelno održavaju zapis transakcija i zapis znanstvenih podataka na istom *blockchainu*. Generiranje novog bloka u slučaju

ove kriptovalute traje otprilike 1 minutu i ne postoji gornja ograda za broj jedinica kriptovalute koji će u budućnosti biti izdan.

Bytecoin

Nastao je 2012. godine i spada u skupinu anonimnih kriptovaluta. Najpoznatija kriptovaluta `Bitcoin` kritizirana zbog toga što korisnici ne mogu ostati potpuno anonimni pri korištenju, već se pomoću big-data analiza može predvidjeti ponašanje korisnika. Partneri novčanici u `Bitcoin sustavu` sadrže adresu korisnika u obliku javnog i tajnog ključa. Kada korisnik obavlja neku transakciju ona sadrži njegov javni ključ, kažemo da je korisnik potpisao transakciju. `Bytecoin` uvodi inovaciju koja se zove *potpisivanje iz prstena*. Za svaku transakciju u `Bytecoin` mreži definira se grupa korisnika, svaki sa svojim javnim i tajnim ključem. Transakcija sadrži javni ključ neke od osoba iz grupe, ali ostatku sustava je nemoguće odrediti koja je to osoba. Razmak između dva generiranja u bloka u sustavu iznosi dvije minute, a ukupna količina izdanih jedinica kriptovalute je ograničena sa 184 bilijuna.

Freicoin

`Freicoin` kriptovaluta je uvedena 2012. godini. Koristi jednaki proof-of-work algoritam poput `Bitcoina` i ima definirano vrijeme uključenja novog bloka svakih 10 minuta. To je primjer kriptovalute koja je prva uvela negativnu kamatnu stopu na pohranjivanje vrijednosti. Od vrijednost pohranjene u `Freicoin`-ovima oduzima se 4.5% kamata kako bi se potaknula potrošnja i smanjila štednja odnosno gomilanja novca. Freicon kao valuta nije zaživjela, ali je zanimljiv primjer kako se pomoću kriptovaluta mogu provoditi različite monetarne politike.

4.2.1 Namecoin

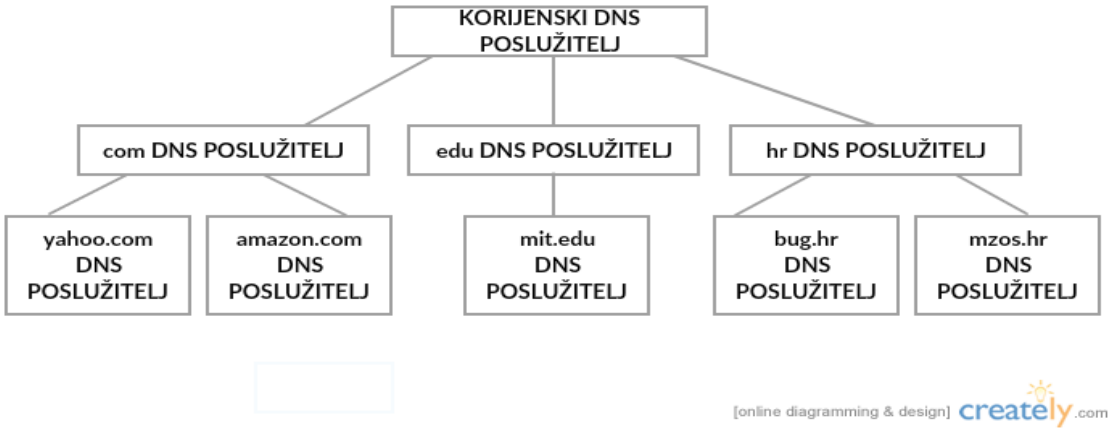
`Namecoin` je također kriptovaluta, no to je zapravo sustav koji koristi *blockchain* tehnologiju kod kojeg kriptovaluta ima sekundarnu ulogu. Zbog toga `Namecoin`-u posvećujemo posebno potpoglavlje.

`Namecoin` je nastao od `Bitcoina` i kao kriptovaluta ima identične karakteristike poput `Bitcoina`, zapravo te dvije valute koriste jednak programerski kod u pozadini. S druge strane `Namecoin` predstavlja distribuiranu platformu za registraciju, koja podržava zapise ključ-vrijednost oblika. Korisnici na takav *blockchain* mogu zapisati informacije poput email adresa, kriptografskih ključeva, SSL certifikata, digitalnih potpisa datoteka (takvi potpisi omogućuju identifikaciju i verifikaciju sadržaja računalnih datoteka) i još mnogo toga. Najvažnija uloga `Namecoina` *blockchaina* je

održavanje skupa zapisa registriranih domenskih imena (*eng. domain-name*) baš poput distribuiranog sustava za registraciju domenskih imena koje koristi Internet.

`Namecoin sustav` se koristi kao alternativa postojećoj centraliziranoj *Domain Name System* (DNS) aplikaciji, s *vršnom* domenom naziva `.bit`. Pogledajmo najprije kako funkcionira centralizirana aplikacija DNS.

Centralizirani DNS je primjer aplikacije građene prema modelu *klijent-poslužitelj*. DNS poslužitelj prevodi simbolička imena računala u IP adrese. Na primjer kada u web preglednik upišemo google.com, DNS poslužitelj našeg računala vraća IP adresu 173.194.70.113. Poslužitelji u tom sustavu kao i simbolička imena računala strukturirani su hijerarhijski. Dijelovi hijerarhije imena računala zovu se domene.



Slika 4.1: Hijerarhija DNS poslužitelja

Poslužitelji na drugoj razini stabla sa slike 4.1 nazivaju se vršni poslužitelji. Tim poslužiteljima upravlja ICAAN (engl. Internet Corporation for Assigned Names and Numbers). Između ostalog ICAAN-a dodjeljuje nazive domena organizacijama koje se nalaze niže u hijerarhiji. Te organizacije nazive vršnih domena opet mogu dodijeliti ili prodati ostalim niže rangiranim organizacijama. Takvim načinom dolazimo npr. do imena `math.pmf.unizg.hr`. U vršne domene spadaju tzv generičke domene `.com`, `.info`, `.edu`, `.org` ili nacionalne domene (engl. ccTLD = Country Code Top-Level Domain), npr. `.hr`, `.au`, `.rs`, `.it`.

Prevođenje simboličkih imena kreće od lokalnog DNS poslužitelja. Ako on u svojoj bazi podataka nema zapisan podatak koji se od njega traži, taj poslužitelj postaje

klijent i povezuje se vršnim poslužiteljem ili nekim od poslužitelja koji su odgovorni za pod-domene.

Kao što smo vidjeli DNS poslužiteljima upravljaju velike organizacije i vlade država. One mogu zlorabiti svoju moć tako da cenzuriraju i nadziru naše korištenja Interneta. To se događalo diljem svijeta kroz povijest, a događa se i danas.

`Namecoin` je uz pomoć *blockchain tehnologije* koju koristi imun na ovakve probleme. Umjesto klijent-poslužitelj modela za građu DNS sustava `Namecoin` koristi decentralizirani model ravnopravnih partnera. Baze podataka koje povezuje simbolička imena računala s IP adresama u ovom slučaju nisu u vlasništvu neke organizacija ili vlade, već takve zapise svaki partner u sustavu samostalno održava pomoću *blockchaina*. *Blockchain tehnologija* omogućava da konzistentnost tih zapisa i nemogućnost mijenjanja istih.

Nabrojimo neke prednosti decentraliziranog DNS sustava u odnosu na prije opisani centralizirani:

- Registracija domene znatno je jeftinija korištenjem decentraliziranog sustava temeljenog na *blockchain tehnologiji*.
- Decentralizirani sustav poštuje privatnost korisnika. Sustav ne zahtijeva od korisnika pružanje identifikacijskih podataka koje kasnije može koristiti protiv njih. Na primjer, kod centraliziranog DNS sustava vlada ima mogućnost kontrole neovisnih medija.
- Vlade ili organizacije mogu blokirati ili oduzeti ime nekoj regularnoj domeni, *blockchain tehnologija* to onemogućava.
- Novac potrošen na registraciju domene korisnika pomaže pri održavanje i daljnjem razvoju drugih decentraliziranih javnih usluga.
- Decentralizirani DNS sustav omogućuje postojanje vršne domena koje nije u ničijem vlasništvu (`.bit` domena). Dokle god postoje partneri odnosno čvorovi u mreži koji obavljaju ulogu DNS poslužitelja svi ostali korisnici imaju pristup bilo kojoj domeni po hijerarhiji nižoj od `.bit` domene. Takvi partneri su zapravo blockchain partneri sa slike 2.5. Čak i gašenjem poslužitelja niže hijerarhijske razine, vlasti ne mogu nametnuti pravila i utjecati na rad domene najviše razine.

`Namecoin` kriptovaluta (oznaka za jedinicu kriptovalute NMC) je dio `Namecoin sustava`. Koristi se za plaćanje naknade za registraciju i prijenos imena. `Namecoin sustav` podržava nekoliko vrsta transakcija čija je svrha pohranjivanje podataka u *blockchain*. Pregled tih transakcija dan je u tablici 4.1, na cijenu transakcije dodaje se i naknada od 0.005 NMC. Ograničenje na ukupan broj izdanih NMC-ova je 21

milijun, kreiranjem svakog novog bloka rudar je nagrađen s 50 novonastalih jedinica kriptovalute `Namecoin`.

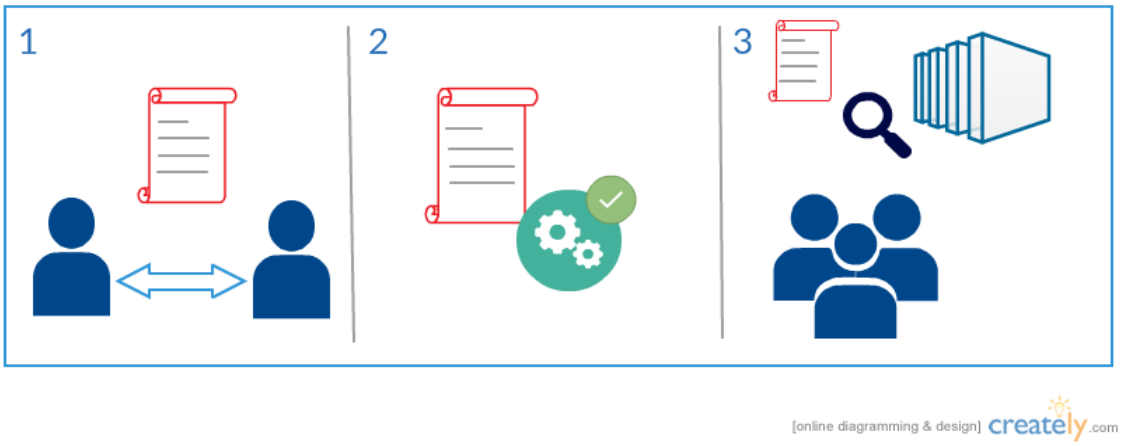
Tablica 4.1: Transakcije u Namecoin sustavu

Naziv	Cijena	Opis
name_new	0.01 NMC	Omogućuje korisniku da se predbilježi za neku domenu
name_firstupdate	0 NMC	Transakcija registrira ime i čini ga javnim
name_update	0 NMC	Omogućuje mijenjanje ili obnavljanje domene

4.3 Pametni ugovori

Pametni ugovor predstavlja kod u nekom programskom jeziku koji olakšava razmjenu novca, nekretnina, dionica ili bilo kakvih vrijednosti. Možemo reći da pametni ugovori služe za reguliranje nekog poslovnog odnosa između stranaka među kojima ne postoji uzajamno povjerenje. Takav kod se može zapisati na *blockchain* i izvršavati na bilo kojem računalu u distribuiranoj mreži. Pametan ugovor se automatski izvršava kada su zadovoljeni specifični uvjeti. Zbog činjenice da je kod pametnog ugovora zapisan na *blockchain* izvršavanje se odvija bez ikakve mogućnosti cenzure, stanke, prijevare ili uplitanja treće strane. Možemo reći da *blockchain* na kojem su pohranjeni pametni ugovori predstavlja distribuirani operacijski sustav.

Imovina ili valuta prenosi se u program. U trenutku izvršavanja programa on automatski prepoznaje stanje u kojem se sustav nalazi. Također, kada više osoba zatraži trajno ili privremeno vlasništvo nad tim vrijednostima program određuje kome one trebaju pripasti. Postoji mogućnost da nitko ne zadovoljava uvjete, tada se vrijednosti vraćaju početnom vlasniku. U međuvremenu, neka vrsta dokumenta u kojem je zapisana odluka programa se pohranjuje *blockchain*, koji mu daje određenu sigurnost i nepromjenljivost.



Slika 4.2: Pametni ugovor

Slika 4.2 prikazuje proces kreiranja pametnih ugovora. Koraci su sljedeći:

1. Moguća razmjena dobara između dva (ili više) partnera zapisuje se kao programerski kod i pohranjuje na *blockchain*. Partneri ostaju anonimni, no sadržaj pametnog ugovora javno je dostupan svim partnerima u sustavu.

2. Varijable poput datuma ili određene količine novca potiču izvršavanje ugovora prema pravilima definiranim u kodu.

3. Ostali korisnici sustava mogu pretražiti *blockchain*, kako bi razumjeli aktivnosti definirane ugovorom ili provjerili rezultat izvršavanja ugovora.

Zamislimo da osoba A želi gostovati u apartmanu koji iznajmljuje osoba B. Osoba B može transakcijom upisanom u *blockchain* platiti svotu potrebnu za iznajmljivanje apartmana. Tako dobiva digitalni račun koji je sadržan u virtualnom ugovoru između te dvije stranke. Osoba A tada šalje osobi B digitalni ključ koji će osobi B biti na raspolaganju od dogovorenog datuma. Ako B ne primi ključ na vrijeme, kod koji sadrži pametni ugovor automatski vraća uplaćena sredstva. Ako A prerano pošalje ključ, funkcija unutar programa ga čuva do datuma kada je dogovoreno iznajmljivanje. Zajedno s ključem funkcija čuva i naknadu u kriptovaluti koju isplaćuje osobi A kada osoba B primi ključ. Kod je zapisan u *blockchain* koji održava na tisuće partnera u sustavu i B ne mora brinuti o tome da će doći do greške ili prijevare. Također, osoba A može biti sigurna da će mu usluga iznajmljivanja biti plaćena ako pošalje ključ. Ugovor se automatski isključuje nakon dogovorenog vremena, a kod ne može

biti mijenjan od strane bilo kojeg sudionika bez znanja drugoga, svi će sudionici biti istovremeno upozoreni o promjenama.

Na primjeru iznajmljivanja apartmana vidjeli smo samo jednu od mogućih uporaba pametnih ugovora. Nabrojimo još neke slučajeve u kojima se koriste ili postoji mogućnost da će se u budućnosti koristiti pametni ugovori:

- automatizacija sustava glasovanja, gdje *blockchain tehnologija* može pomoći pri vjerodostojnosti cijelog sustava,
- klinička istraživanja provedena od strane više institucija uz zaštitu osobnih podataka ispitanika
- praćenje udjela vlasništva i investicija od strane kompanije koja radi na projektu u koji sredstva ulažu strani ulagači,
- automatizacija isplate i vraćanja kredita kao i praćenje kamata

itd.

U kombinaciji s drugim tehnologija dobivamo još širu uporabu pametnih ugovora. Na primjer, možemo automatizirati osiguranje vozila i omogućiti gotovo trenutnu isplatu osiguravajuće kuće oštećenoj stranci. U slučaju nesreće isplata bi se vršila na temelju podataka prikupljenih od strane senzora koji prate parametre stanja vozila u pametnim automobilima.

4.3.1 Ethereum

Ethereum je najpoznatiji primjer decentraliziranog distribuiranog sustava koji omogućuje izvršavanje pametnih ugovora koristeći *blockchain tehnologiju*. U sklopu Ethereum sustava koristi se kriptovaluta Ether. Ether ima dvojaku ulogu, koristi se pri plaćanju postavljanja ugovora na mrežu ali i kao sredstvo za isplaćivanja nagrade partnerima rudarima pri kreiranju novog bloka, baš kao u `Bitcoin sustavu`.

Pogledajmo kako Ethereum funkcionira u kratkim crtama, koristeći terminologiju sa slike 2.5. Partneri novčanici ili potpuni partneri kreiraju zapise koji sadrže pametne ugovore. Ugovori se izvršavaju kada se postignu određeni uvjeti. Partneri rudari održavaju skup neriješenih zapisa, koji uključuje postojeće ugovore koji se trebaju izvršiti i ugovore koji se tek trebaju zapisati u *blockchain*. Pri kreiranju novog bloka odabiru neke od tih zapisa i na svojim računalima izvršavaju kod postojećih pametnih ugovora. Izvršavanje koda mijenja stanje računa partnera novčanika na način da se automatski izvršava transakcija kojom neki korisnik uplaćuje drugom korisniku određeni broj jedinica kriptovalute Ether koji je definiran u pametnom ugovoru. Izvršavanje također ima za rezultat neki dokument koji vežemo uz pametni ugovor, te

podatke je potrebno dodati u novi blok. Nakon što rudar zapiše rezultantne podatke zajedno s podacima o novim ugovorima u blok kreće u izvršavanje proof-of-work algoritma za taj blok. Ako nađe broj *nonce* koji daje zadovoljavajući hash difuzijskom porukom šalje novi blok kroz cijelu mrežu.

Bilo koji drugi partner koji održava *blockchain* nakon primitka novog bloka ponovo izvršava kod pametnih ugovora u onom redoslijedu koji je zapisan u bloku koji je primio. Uz to bilježi i provjerava rezultate izvršavanja pametnih ugovora. Također, partner provjerava daje li broj nonce koji je rudar pronašao zaista zadovoljavajući hash i ako je prihvaća taj blok i zapisuje ga u svoju lokalnu kopiju *blockchaina*.

Vidimo da se svi partneri u sustavu moraju složiti oko rezultata izvršavanja pametnih ugovora, tako što na svojim računalima pokreću kod. Zbog toga stranke koje su svoju suradnju zapisali u pametni ugovor mogu vjerovati sustavu.

Poglavlje 5

Web aplikacije Login i Jednostavni novčanik

5.1 Uvod

Pojavom *blockchain* tehnologije razvijeno je i mnoštvo open-source projekata koji omogućuju programerima razvoj aplikacija kao što su pametni ugovori ili implementacije partnera novčanika. Ti open-source projekti također omogućuju komunikaciju aplikacije s nekim javnim *blockchainom* ili izradu svog privatnog *blockchaina*. U sklopu ovog diplomskog rada izrađene su dvije web aplikacije koje će biti opisane u ovom poglavlju. To su Jednostavni Bitcoin novčanik i Login aplikacija.

5.2 Implementacija Jednostavnog novčanika

Novčanik u `Bitcoin sustavu` je računalni program koji služi za primanje i slanje bitcoina. Uloga novčanika je pohrana privatnih ključeva korisnika, prikaz transakcija korištenih za slanje ili primanje bitcoina pomoću adrese korisnika te prikaz količine bitcoina koje korisnik posjeduje. Bitcoine iz jednog novčanika u drugi prebacujemo `Bitcoin` transakcijama. Novčanik je moguće instalirati na računalo, pametni telefon ili tablet. Oni su također dostupni kao web aplikacije, kojima je moguće pristupiti sa svakog uređaja povezanog na internet.

Upoznajmo najprije najvažnije pojmove za funkcioniranje novčanika. To privatni ključ, javni ključ, adresa i transakcija. **Privatni ključ** je jednostavno rečeno broj koji možemo generirati na razne načine. Da bismo mogli definirati vezu između privatnog i javnog ključa definirajmo najprije *eliptičku krivulju*, vrstu krivulje često korištenu kriptografiji.

Definicija 5.2.1. *(Eliptička krivulja)*
Neka je K polje karakteristike različite od 2 i 3, te neka je $f(x) = x^3 + ax + b$ (gdje su $a, b \in K$) kubni polinom bez višestrukih korijena. Eliptička krivulja *E nad K je skup svih točaka $(x, y) \in K \times K$ koje zadovoljavaju jednadžbu $y^2 = x^3 + ax + b$, zajedno s još jednim elementom kojeg označavamo s O i zovemo "točka u beskonačnosti".*

Polje je algebarska struktura, a karakteristika polja K je najmanji prirodni broj n takav da je $1 + 1 + ... + 1 = n \cdot 1 = 0$, gdje su 0 i 1 neutralni elementi za zbrajanje, odnosno množenje u K. Eliptičke krivulje se koriste pri generiranju javnog ključa iz privatnog.

Najčešće se generiranje privatnog ključa svodi na nasumični odabir broja između 1 i 2^{256} na koji primjenjujemo SHA-256 hash funkciju. Jednom kad smo generirali privatni ključ možemo pomoću njega i eliptičke krivulje izračunati **javni ključ**. `Bitcoin sustav` koristi *eliptičku krivulju* zadanu jednadžbom

```
(y^2 = x^3 + 7) mod p,
```

gdje je

```
p= 2^256 - 2^32 - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1.
```

Uz zadanu krivulju vežemo točku G koja se naziva generator eliptičke krivulje. Ukoliko privatni ključ označimo s pk tada pripadajući javni ključ jk računamo po formuli

```
jk = pk*G.
```

Dakle javni ključ jk je točka na krivulji dobivena matematičkom operacijom množenja točke G skalarom pk što je zapravo zbrajanja točke G sa samom sobom pk puta. Nakon toga pomoću kompozicije nekoliko hash funkcija, među kojima je i SHA-256 funkcija iz javnog ključa možemo računamo `Bitcoin` adresu A. Ako kompoziciju hash funkcija označimo s $\mathtt{H}_{\mathtt{adr}}$, adresu A računamo prema sljedećoj formuli:

$$\mathtt{A} = \mathtt{H}_{\mathtt{adr}}(\mathtt{jk}).$$

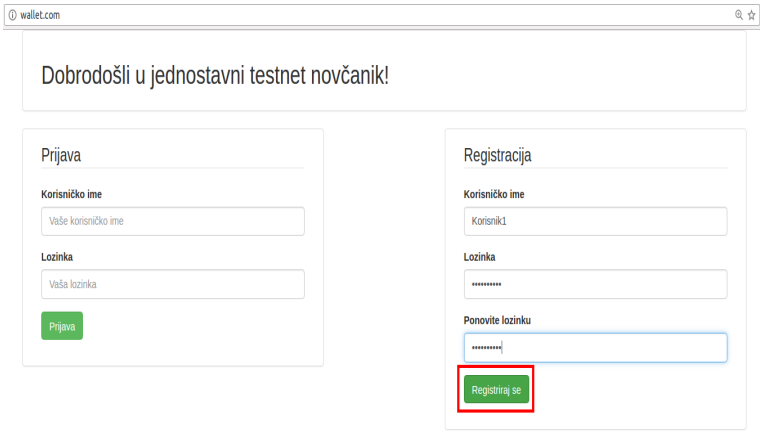
Privatni ključ koristi se pri kreiranju transakcije. Može biti pohranjen u novčaniku u različitim formatima. WIF (Wallet Import Format) je jedan od najčešćih formata zapisa privatnog ključa. Pomoću privatnog ključa korisnik potvrđuje da je upravo on vlasnik određene adrese. Stoga možemo reći da bitcoini na nekoj adresi pripadaju osobi koja posjeduje privatni ključ za tu adresu. Važno je napomenuti da su operacija množenja na eliptičkoj krivulji i hash funkcije jednosmjerne odnosno nemaju inverz. Tako je u svakom trenutku lako iz privatnog ključa generirati javni i iz javnog ključa

generirati adresu. No, ne postoji način da se pomoću adrese generira privatni ili javni ključ.

Transakcija je zapis u `Bitcoin` mreži kojim se određeni iznos bitcoina prenosi s jedne adrese (ili više njih) na drugu adresu (ili više njih). Transakcije su javne i moguće ih je slobodno pregledavati, primjerice pomoću *blockchain* pretraživača dostupnih na internetu. One nedvosmisleno potvrđuju da je određena količina bitcoina prenesena s jedne adrese na drugu. Novčanik omogućava korisniku da kreira novu transakciju s adrese koja pripada tom novčaniku. Da bi se transakcija izvršila potrebno je da korisnik navede adresu primatelja i iznos koji želi uplatiti. Uz transakciju vežemo ulaz, koji se sastoji od prethodnih transakcija uplaćenih na adresu korisnika. Posjedovanjem privatnog ključa korisnik potvrđuje da određena svota novca pripada njemu i da te transakcije može koristiti kao ulaz za novu transakciju. Svaka transakcija ima i svoj izlaz, pomoću kojeg se definira koji se dio od ukupne svote novca prebacuje na adresu primatelja, a koji dio ostaje na adresi pošiljatelja.

Implementacija novčanika u sklopu ovog rada koristi open-source projekte `bitcore` i `bitcore-explorers`. Pomoću *Node.js* JavaScript platforme i modula `bitcore` i `bitcore-explorers` koriste se metode za kreiranje ključeva, stvaranje transakcija, pretraživanje *blockchaina* i slanje transakcija u `Bitcoin` mrežu. `Bitcoin` je za potrebe programera razvio testni sustav pod nazivom *testnet* koji funkcionira na isti način kao i stvarni `Bitcoin sustav` samo što bitcoinovi na toj mreži nemaju vrijednost.

Aplikacija je napravljena na modelu klijet-poslužitelj. Klijent (internet preglednik) komunicira s dva poslužitelja. *Node.js* poslužiteljem koji pruže metode implementirane pomoću `bitcore` i `bitcore-explorers` modula i Apache poslužiteljem na koji sprema podatke o svojim korisnicima. Da bi se koristile usluge aplikacije korisnik se najprije mora registrirati. Formu za registraciju možemo vidjeti na slici 5.1.



Slika 5.1: Registracija u jednostavni novčanik

Korisnik unosi podatke za registraciju. Pritiskom na gumb *Registriraj se* na strani klijenta obavlja se validacija podataka. Ako su podaci ispravni klijent šalje zahtjev pomoću metoda klase $XMLHttpRequest()$; i traži od *Node.js* poslužitelja generiranje novog privatnog ključa. Kod za generiranje novog ključa na strani *Node.js* poslužitelja izgleda ovako:

```
1 var bitcore = require('bitcore');
2 var rand_buffer = bitcore.crypto.Random.getRandomBuffer(32);
3 var rand_number = bitcore.crypto.BN.fromBuffer(rand_buffer);
4 var privateKey = new bitcore.PrivateKey(rand_number,'testnet');
5 var privateKeyWif = privateKey.toWIF();.
```

Isječak koda 5.1: Generiranje novog privatnog ključa

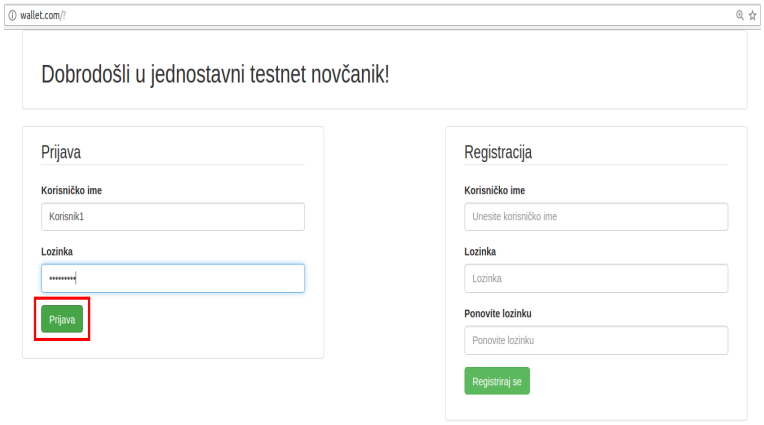
Po primitku novog privatnog ključa klijent registrira novog korisnika na bazi. Podaci koji se spremaju na bazu su korisničko ime, hash korisničkog imena i lozinke dobiven SHA-256 algoritmom i kriptirani javni ključ potpisan s lozinkom pomoću AES (Advanced Encryption Standard) kriptografskog algoritam za zaštitu digitalnih podataka. Kriptirani javni ključ i hash dobivamo pomoću sljedećeg isječka JavaScript koda na strani klijenta:

```
1 var hash = asmCrypto.SHA256.hex(uname.value+pass.value);
2
3 var key = aesjs.util.convertStringToBytes(pass.value);
4
```

```
5  //privateKeyWif dobiven od strane Node.js posluzitelja
6  var textBytes = aesjs.util.convertStringToBytes(privateKeyWif);
7
8  var aesCtr = new aesjs.ModeOfOperation.ctr(key, new aesjs.Counter(5)
       );
9  var encryptedBytes = aesCtr.encrypt(textBytes);
10 var encryptedText = aesjs.util.convertBytesToString(encryptedBytes);
```

Isječak koda 5.2: Enkripcija javnog ključa i SHA-256 hash

Ako je registracija uspješno prošla. Korisnik je spreman za prijavu.



Slika 5.2: Prijava u jednostavni novčanik

Nakon popunjavanja podataka, pritiska na gumb *Prijava* klijent traži od Apache poslužitelja podatke o tom korisniku. Ponovo računa hash pomoću upisanog korisničkog imena i lozinke i ako se hash vraćen s baze podataka podudara s izračunatim korisnik može početi s korištenjem aplikacije. U ovom trenutku klijentu je potrebna `Bitcoin` adresa korisnika. Sada pomoću lozinke dekriptira privatni ključ:

```
1  key = aesjs.util.convertStringToBytes(pass.value);
2  var aesCtr = new aesjs.ModeOfOperation.ctr(key, new aesjs.Counter(5)
       );
3
4  //encriptedData podatak iz baze
5  var decryptedBytes = aesCtr.decrypt(encriptedData);
6
```

```
7 var pk = aesjs.util.convertBytesToString(decryptedBytes);.
```

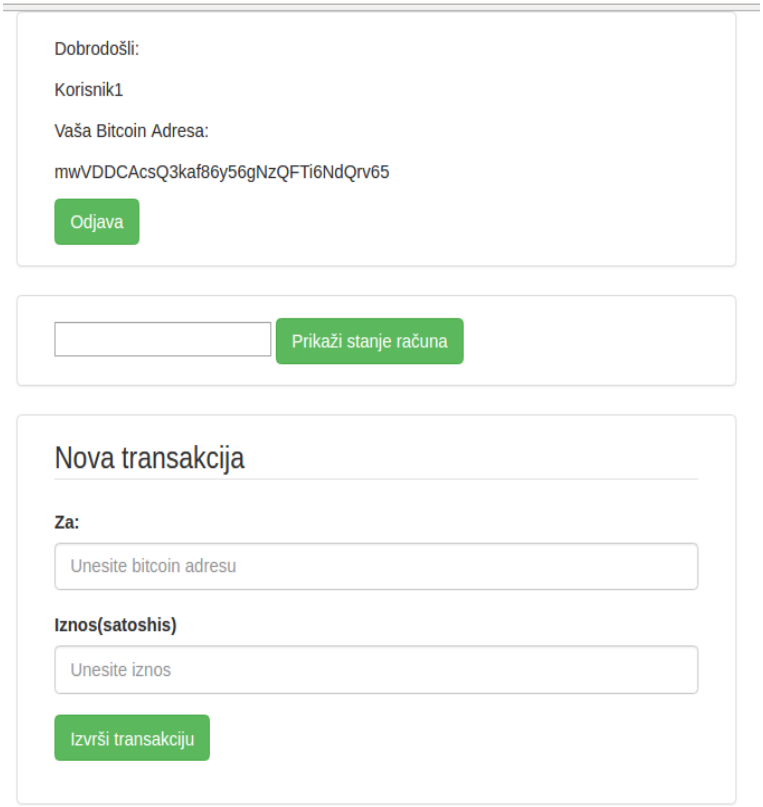
Isječak koda 5.3: Dekripcija javnog ključa

Pomoću privatnog ključa *pk* i klase $XMLHttpRequest()$; traži se adresa koja odgovara tom privatnom ključu. Adresu generirana metodama bitcore modula:

```
1 var bitcore = require('bitcore');
2 //pk primljen od klijenta
3 var privateKey = bitcore.PrivateKey.fromWIF(pk);
4 var add = privateKey.toAddress();.
```

Isječak koda 5.4: Generiranje Bitcoin adrese

Sučelje aplikacije možemo vidjeti na slici 5.3. Korisnik može pomoću sučelja doznati stanje svog računa ili izvršiti novu transakciju.



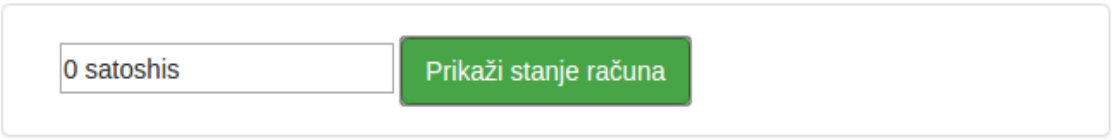
Slika 5.3: Sučelje aplikacije jednostavni novčanik

Stanje računa je u aplikaciji prikazano u satoshi jedinicama kriptovalute `Bitcoin`. Vrijedi 1 BTC = 100,000,000 satoshi i to je najmanja jedinica na koju se dijeli valuta. Stanje računa je jedna od svojstava svake adrese u `Bitcoin sustavu` i pomoću metode `address bitcore-explorers` modula dobivamo ga na sljedeći način:

```
1 var Insight = require('bitcore-explorers').Insight;
2 var insight = new Insight('testnet');
3 insight.address(body.trim(), function(err, addInfo) {
4   var stanjeRacuna = addInfo['balance']+' satoshis';
5 });.
```

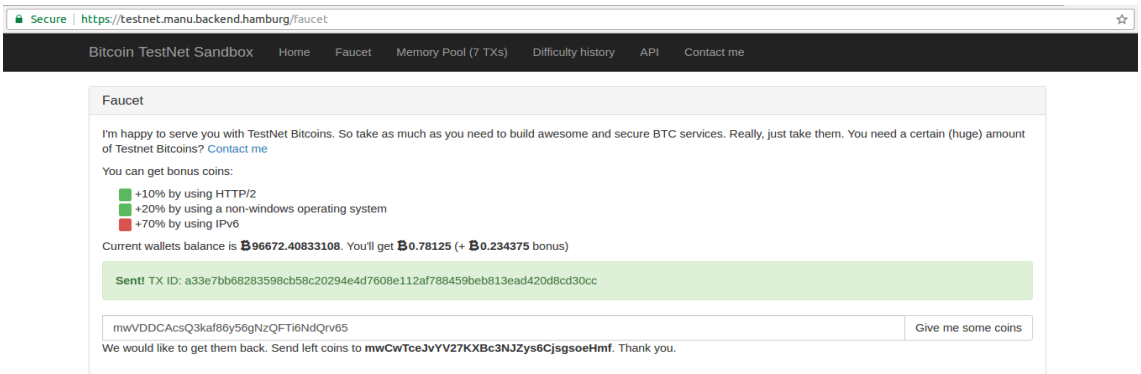
Isječak koda 5.5: Provjera stanja računa

Korisnik1 je novi korisnik `Bitcoin` testnet mreže i ne postoji ni jedna izvršena transakcija kojom bi se na njegovu adresu uplatio neki iznos. Pritiskom na gumb *Prikaži stanje računa* korisnik dobiva informaciju da stanje računa iznosi 0 satoshi kao što možemo vidjeti na slici 5.4.



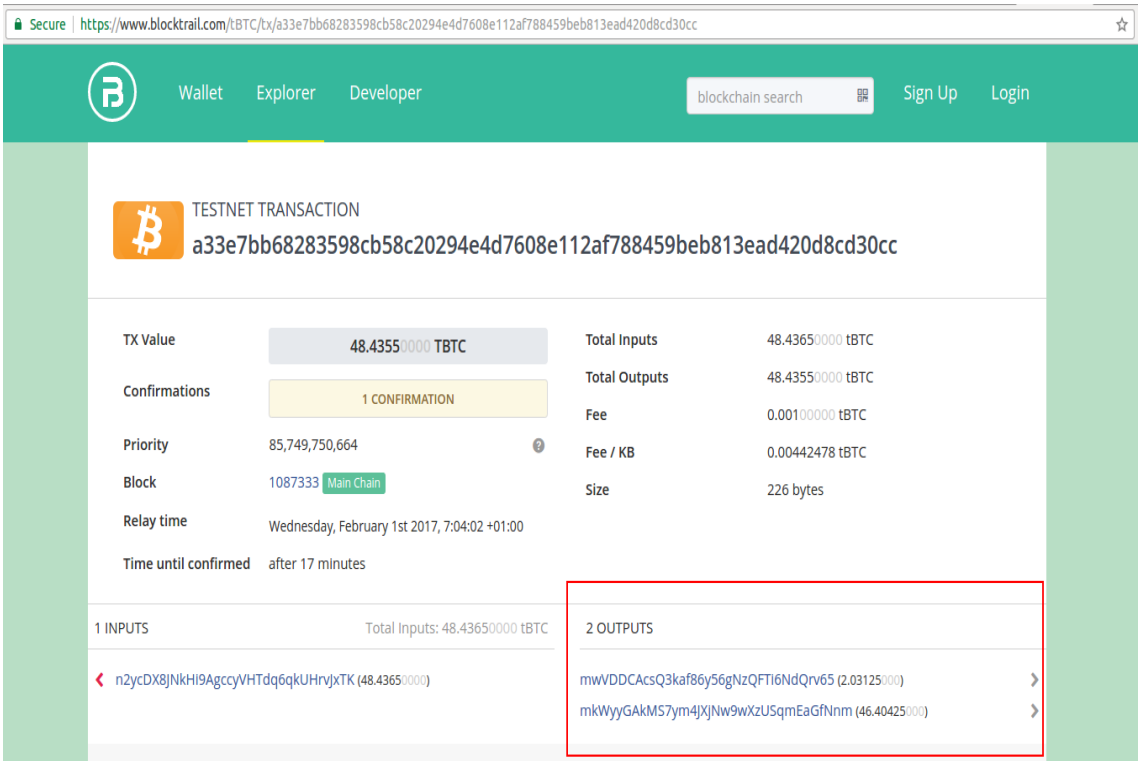
Slika 5.4: Stanje računa novog korisnika

Budući da je `Bitcoin` testnet mreža osmišljena za razvoj i testiranje aplikacija i da novac na toj mreži nema nikakvu vrijednost postoje internetske stranice koje kreiraju transakciju s adresa koje posjeduju testnet novac na vašu adresu. To je način da *Korisnik1* dođe do svojih prvih bitcoinova. Na slici 5.5 prikazan je zahtjev za dobivanjem testnet bitcoinova na adresu `mwVDDCAcsQ3kaf86y56gNzQFTi6NdQrv65` koja pripada korisniku *Korisnik1*.

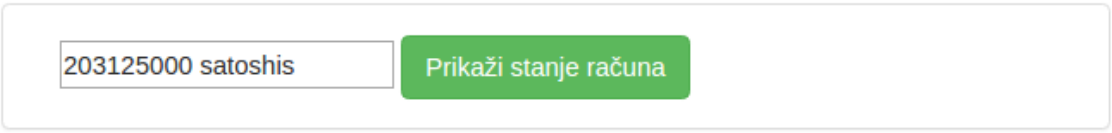


Slika 5.5: Dobivanje prvih bitcoinova

Prva transakcija na adresu korisnika *Korisnik1* je izvršena. Na slici 5.5 vidimo da je broj `a33e7bb68283598cb58c20294e4d7608e112af788459beb813ead420d8cd30cc` *id* te transakcije. Pomoću tog podatka možemo korištenjem bilo kojeg testnet *blockchain* pretraživača dostupnog na internetu pronaći dodatne informacije o transakciji. Transakciju možemo pronaći gotovo istog trenutka kada je kreirana. No, prisjetimo se da je zbog proof-of-work algoritma koji koristi `Bitcoin sustav` potrebno neko vrijeme da transakcija bude potvrđena odnosno upisna u novi blok. Slika 5.6 prikazuje informacije o transakciji dobivenih pomoću Blocktrail testnet *blockchain* pretraživača. Primijetimo da je ukupan ulaz transakcije nešto već od ukupnog izlaza. Razlog tome je taj što se na svaku transakciju pridodaje naknada koja će kasnije biti dodijeljena rudaru koji zapisuje novi blok u *blockchain*. Transakcija sa slike sadrži dva izlaza i je jedan od njih uplaćen na adresu `mwVDDCAcsQ3kaf86y56gNzQFTi6NdQrv65`. Ako sada ponovo u našoj novčanik aplikaciji zatražimo prikaz stanja korisnik *Korisnik1* dobivamo rezultat sa slike 5.7.

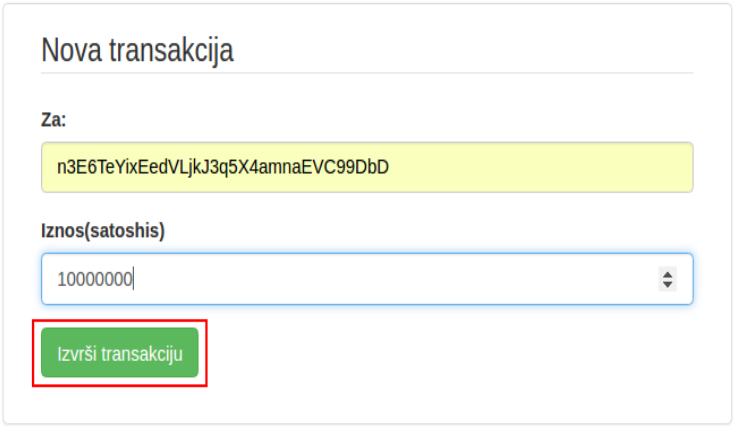


Slika 5.6: Prikaz transakcije u testnet blockchain pretraživaču



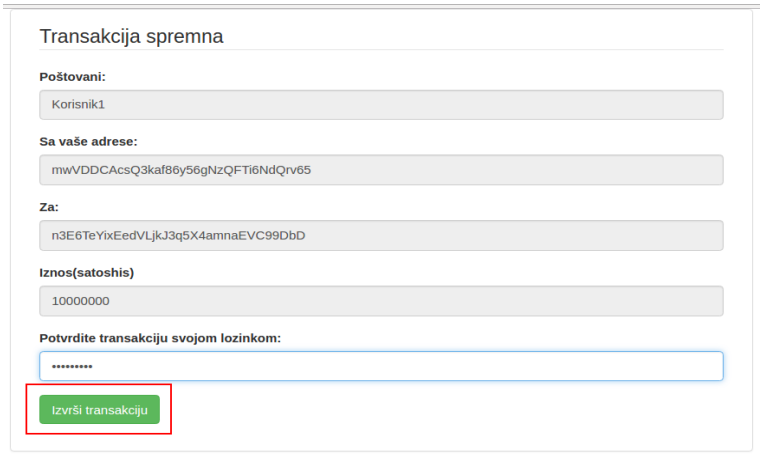
Slika 5.7: Stanje računa korisnika nakon prve transakcije

Korisnik1 je sada vlasnik 203125000 testnih satoshija odnosno 2,03125 testnih BTC-a. Preostaje nam pokazati kako aplikacija Jednostavni novčanik kreira novu transakciju. Pretpostavimo da, iz nekog razloga *Korisnik1* želi uplatiti iznos od 10000000 satoshi (0,1 BTC) na adresu `n3E6TeYixEedVLjkJ3q5X4amnaEVC99DbD`. On tada popunjava formu u aplikaciji na način na koji je prikazano na slici 5.8 i pritišće gumb *Izvrši transakciju*.



Slika 5.8: Kreiranje nove transakcije

Aplikacija nakon toga preusmjerava korisnika na novu formu sa slike 5.9. Od korisnika se traži da potvrdi svoju transakciju lozinkom. *Korisnik1* provjerava jesu li podaci točni, pa upisuje lozinku i pritiskom na gumb *Izvrši transakciju* kreira novu transakciju na `Bitcoin` testnet mreži.



Slika 5.9: Potvrda nove transakcije

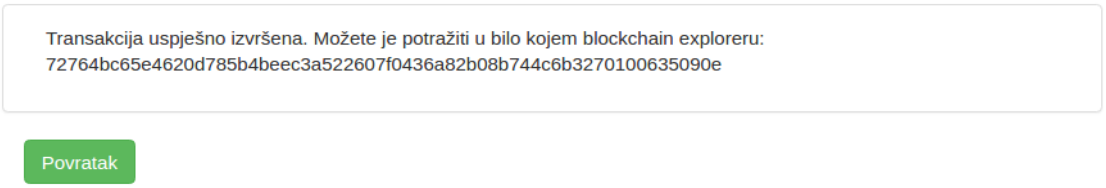
Klijentski dio aplikacije nakon što je korisnik ponovo upisao svoju lozinku može dekriptirati privatni ključ korisnika i poslati zahtjev za kreiranjem nove transakcije *Node.js* poslužitelju. Podaci koje mora priložiti uz taj zahtjev su adresa pošiljatelja, adresa primatelja, iznos transakcije i privatni ključ pošiljatelja. Pogledajmo kako izgleda isječak koda na strani *Node.js* poslužitelja koji kreira transakciju i šalje je difuzijom (broadcast) u `Bitcoin` mrežu. Za izvršenje tog koda opet su potrebni moduli `bitcore` i `bitcore-explorers`. Svaka transakcija kako ulaz prima prijašnje transakcije pošiljatelja, pa je najprije potrebno pretražiti *blockchain* i pronaći takve transakcije metodom `getUnspentUtxos`. Nakon toga se zadaju iznos i adresa primatelja i transakcija se potpisuje privatnim ključem pošiljatelja.

```
1  var Insight = require('bitcore-explorers').Insight;
2  var insight = new Insight('testnet');
3  var bitcore = require('bitcore');
4  insight.getUnspentUtxos(parsedData['addPos'], function(err,trPos){
5    var tx =bitcore.Transaction();
6    tx.from(trPos);
7    tx.to(parsedData['addPrim'], parseInt(parsedData['iznos']));
8    tx.change(parsedData['add']);
9    tx.sign(parsedData['privateKey']);
10   tx.serialize();
11
12   insight.broadcast(tx,function(err, returnedTxId){
13     var toReturn = 'Transakcija uspjesno izvrsena. '
14         + 'Mozete je potraziti u bilo kojem blockchain exploreru: '
```

```
15          + returnedTxId;
16
17      });
18
19  });.
```

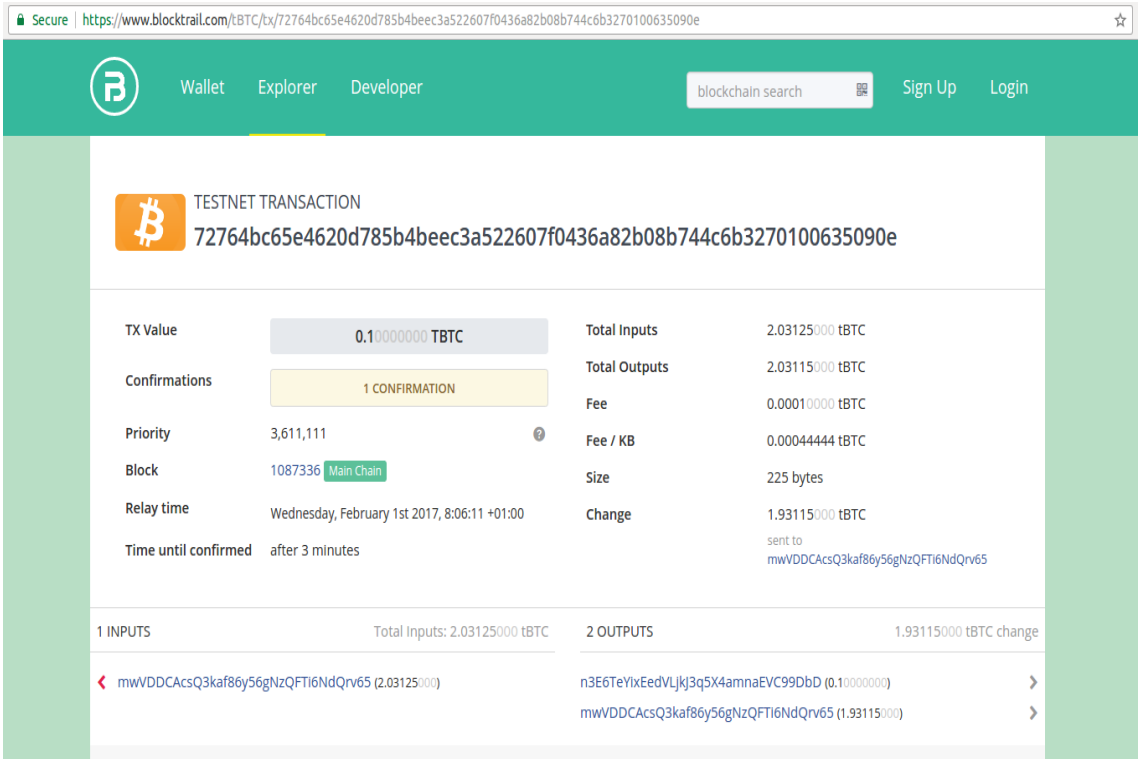
Isječak koda 5.6: Kreiranje nove transakcije

Ako je sve prošlo bez greške poslužitelj će klijentu vratiti poruku koja sadrži *id* nove transakcije. Klijent tu poruku ispisuje korisniku(slika 5.10).

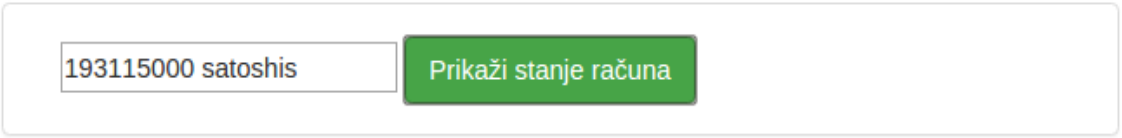


Slika 5.10: Uspješno kreirana transakcija

Zaista, transakcija iz naše aplikacije poslana je na `Bitcoin` testnet mrežu. Što dokazuje slika 5.11 i činjenica da je možemo prikazati u *blockchain* pretraživaču. Također, slika 5.12 prikazuje stanje računa korisnika *Korisnik1* umanjeno je za iznos transakcije i transakcijsku naknadu.



Slika 5.11: Prikaz transakcije u testnet blockchain pretraživaču



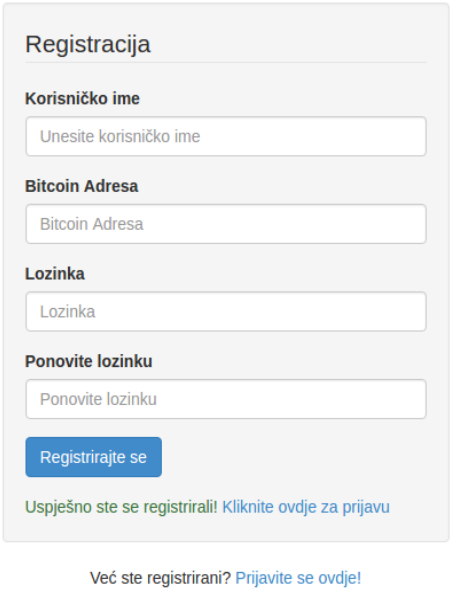
Slika 5.12: Stanje računa korisnika nakon prve transakcije

5.3 Implementacija Login aplikacije

Druga web aplikacija implementirana u sklopu rada je Login aplikacija. Ona također kosti `bitcore` i `bitcore-explorers` *Node.js* module i dodatno modul `blocktrail-sdk`.

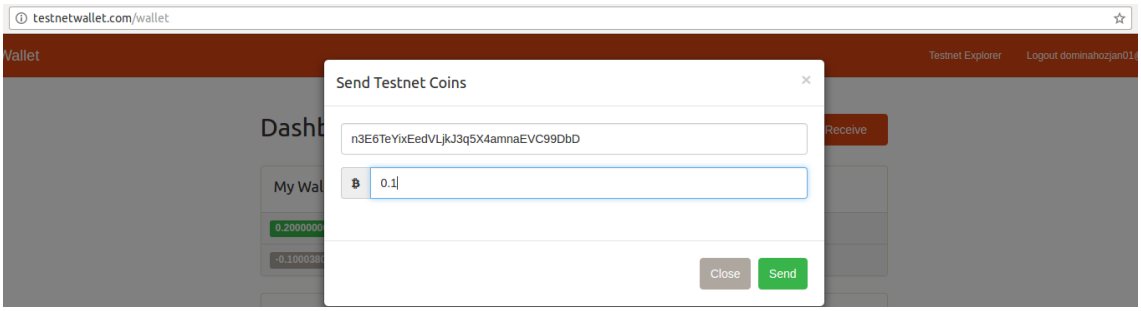
Aplikacija dopušta korisnicima da se prijave u stranicu samo ako su prethodno uplatili iznos od 0.1 BTC na određenu `Bitcoin` adresu. Ta adresa pripada vlasnicima aplikacije koji na njoj objavljuju neki sadržaj. Motivacija za izradu takve aplikacije nađena je u ilegalnim aktivnostima s kojima se `Bitcoin` danas povezuje. Pretpostavimo da je sadržaj same aplikacije ilegalan ali u isto vrijeme za njegovo korištenje je potrebno platiti određeni iznos. Korisnici takvog sadržaja ne žele otkrivati svoj identitet. Bez obavljanja plaćanja pomoću `Bitcoin sustava` lako bi se moglo utvrditi o kojem korisniku se radi budući da bi on uplatom preko banke stavio na raspolaganje svoje podatke. Ova aplikacija omogućuje korištenje sadržaja samo onim korisnicima koji to i platili. S druge strane iskorištena je anonimnosti koju pruža `Bitcoin` u svrhu anonimnosti korištenja aplikacije.

Slika 5.13: Početna stranica Login aplikacije

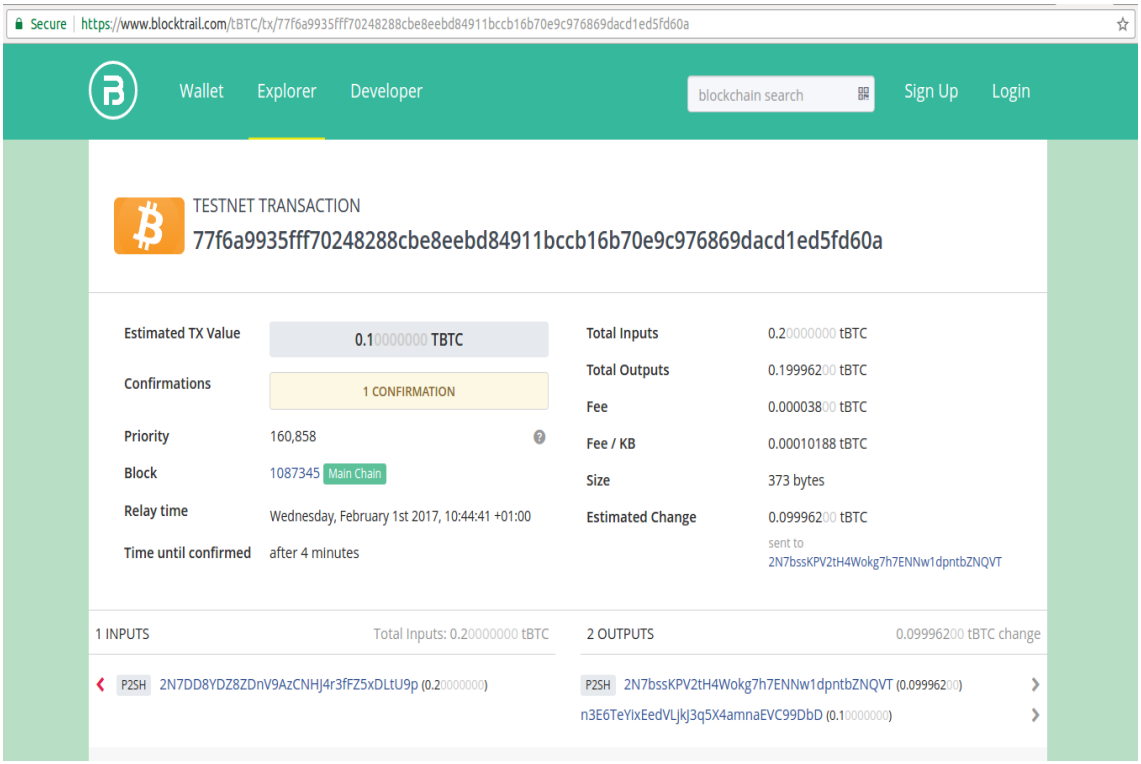


Slika 5.14: Forma za registraciju korisnika

Kako bismo prikazali na koji način aplikacija funkcionira pretpostavimo da imamo 3 registrirana korisnika *Korisnik1*, *Korisnik2* i *Korisnik3*. Registracija se vrši samo putem `Bitcoin` adrese i korisničkog imena kao što je vidljivo na slici 5.14. Početna stranica aplikacije sa slike 5.13 daje informacije o tome na koju je adresu potrebno uplatiti 0,1 BTC za korištenje sadržaja aplikacije. *Korisnik1* je naš korisnik iz prethodnog poglavlja. Registriran je svojom adresom `mwVDDCAcsQ3kaf86y56gNzQFTi6NdQrv65`, mogao je pri registraciji odabrati i neko drugo korisničko ime ali odlučio je koristiti isto. Prisjetimo se da je *Korisnik1* na primjeru kreiranja transakcije uplatio upravo 0.1 BTC na adresu `n3E6TeYixEedVLjkJ3q5X4amnaEVC99DbD`. *Korisnik2* registriran s adresom `2N7DD8YDZ8ZDnV9AzCNHJ4r3fFZ5xDLtU9p` je korisnik Testnet Wallet novčanika (dostupnog na http://testnetwallet.com/ linku). On je također uplatio traženi iznos na adresu `n3E6TeYixEedVLjkJ3q5X4amnaEVC99DbD` pomoću Testnet Wallet novčanika, što možemo vidjeti na slikama 5.15 i 5.16. *Korisnik3* registriran je s valjanom `Bitcoin` adresom `mzuRGyAPEZS5saRveMvd1uv89A96Mb5g5z`, no nije uplatio traženi iznos na traženu adresa.



Slika 5.15: Uplata korisnika Korisnik2 pomoću Testnet Wallet aplikacije



Slika 5.16: Prikaz transakcije u testnet blockchain pretraživaču

Prilikom prijave na stranicu pomoću forme za prijavu klijent aplikacije se spaja na *Node.js* poslužitelj, šalje mu adresu korisnika i očekuje od poslužitelja informaciju

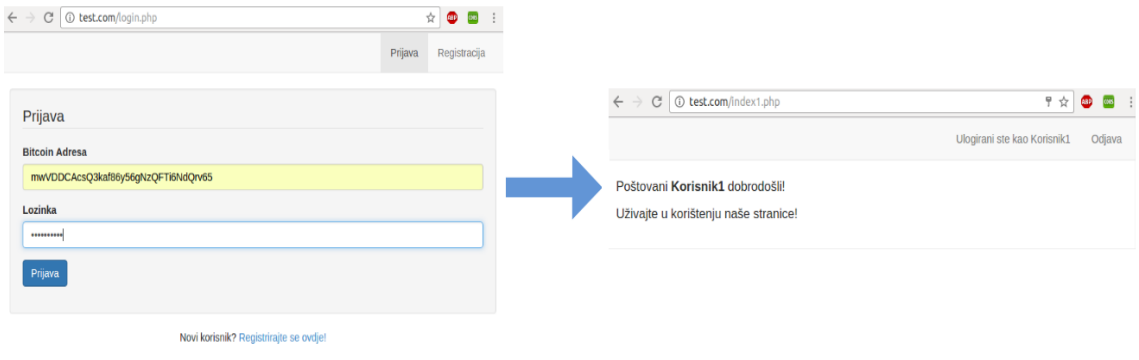
o tome je li korisnik s tom adresom uplatio barem 0.1 BTC na adresu `n3E6TeYixEedVLjkJ3q5X4amnaEVC99DbD`. Slijdi isječak koda koji korištenjem `bitcore`, `bitcore-explorers` i `blocktrail-sdk` modula pretražuje postoji li takva transakcija na *blockchainu*. Najprije se pomoću metode `address` `bitcore-explorers` modula dobivaju sve transakcije vezane uz adresu koju je klijent poslao poslužitelju, transakcije su zapisane u svojstvu `transactionIds` `Bitcoin` adrese. Među svim transakcijama potrebnu je naći onu čiji izlaz sadrži adresu `n3E6TeYixEedVLjkJ3q5X4amnaEVC99DbD` i ima iznos veći ili jednak 10,000,000 sathoshi. Metoda `transaction` modula `blocktrail - sdk` dohvaća informacije o transakciji, među kojima su i izlazi (outputs).

```
var Insight = require('bitcore-explorers').Insight;
var insight = new Insight('testnet');
var bitcore = require('bitcore');
var blocktrail = require('blocktrail-sdk');
var client = blocktrail.BlocktrailSDK({
apiKey: "3210791d4bda9d94546f84e11e88f8179680f929",
apiSecret: "83f67e53a75d21982e129c5fd94331c2ab3827e3",
network: "BTC", testnet: true});
var isPaidToReturn;

//adresaKorisnika primljena od klijenta
insight.address(adresaKorisnika, function(err, addInfo) {
  transactions =addInfo['transactionIds'];
  ForTransactions(transactions, function(err,returned){
    isPaidToReturn=returned;
  });

  function ForTransactions(transactions, done){
    var returned = false;
    var counter = 0;
    for (i = 0; i < transactions.length; i++) {
        IsPaidOutput(transactions[i],function(err,isPaid){
            if(isPaid==true && returned==false){
              done(null,returned);
            }
            if(counter==transactions.length &&
                              isPaid==false && returned==false){
              returned=false;
              done(null,returned);
            }
        });
    }
  }

  function IsPaidOutput(transaction,done){
```

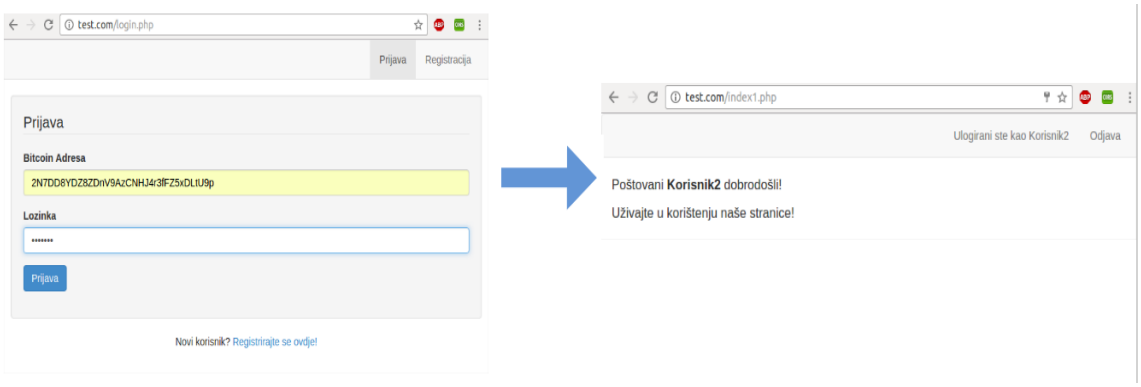
```
36      client.transaction(transactions[i].toString(),function(err, tx){
37        isPaid = false;
38        for (j = 0; j < tx['outputs'].length; j++) {
39          if(tx['outputs'][j]['address']==
40                          'n3E6TeYixEedVLjkJ3q5X4amnaEVC99DbD' &&
41                          parseInt(tx['outputs'][j]['value'])
                               >=10000000){
42            isPaid = true;
43            done(null,isPaid);
44            break;
45           }
46        }
47        done(null, isPaid);
48      });
49    }
50  });
```

Isječak koda 5.7: Provjera je li uplaćen iznos od 0.1 BTC na traženu adresu

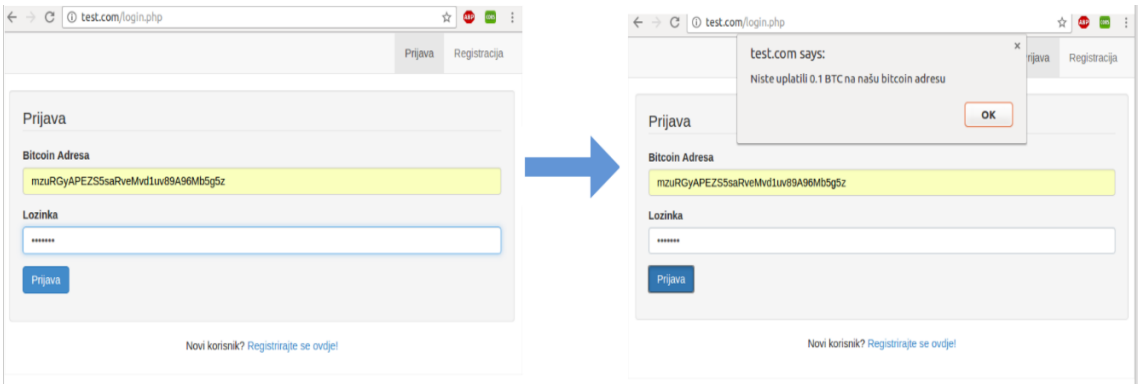
Slike 5.17, 5.18 i 5.19 prikazuju ponašanje aplikacije pri pokušaju prijave korisnika *Korisnik1*, *Korisnik2* i *Korisnik3*. Kao što smo i očekivali prije napisani programerski kod za korisnika *Korisnik3* ne pronalazi traženu transakciju, jer nije nikad uplaćena. Aplikacija korisnicima *Korisnik1* i *Korisnik2* dopušta prijavu i korištenje sadržaja dok se *Korisnik3* ne može prijaviti.



Slika 5.17: Prijava korisnika Korisnik1



Slika 5.18: Prijava korisnika Korisnik2



Slika 5.19: Prijava korisnika Korisnik3

Bibliografija

[1] Andreas M. Antonopoulos, *Mastering Bitcoin*, O'Reilly Media Inc., 2015.

[2] Dug Campbell, *The Byzantine Generals' Problem*, (2015), `http://www.dugcampbell.com/byzantine-generals-problem/`.

[3] Ethereum community, *What is Ethereum?*, (2016), `http://ethdocs.org/en/latest/introduction/what-is-ethereum.html`.

[4] Debraj Ghosh, *How the Byzantine General Sacked the Castle: A Look Into Blockchain*, (2016), `https://medium.com/@DebrajG/how-the-byzantine-general-sacked-the-castle-a-look-into-blockchain-370fe637502c#.939q2cgcq`.

[5] David Gilson, *What are Namecoins and .bit domains?*, (2013), `http://www.coindesk.com/what-are-namecoins-and-bit-domains/`.

[6] Luka Grubišić, Robert Manger, *Mreže Računala*, predavanja, PMF, Zagreb, 2009.

[7] Robert Manger, *Softversko inženjerstvo*, Element, Zagreb, 2011.

[8] Robert Manger, Miljenko Marušić, *Strukture podataka i algoritmi*, predavanja, PMF, Zagreb, 2003.

[9] Rob Marvin, *Blockchain in 2017: The Year of Smart Contracts*, (2016), `http://www.pcmag.com/article/350088/blockchain-in-2017-the-year-of-smart-contracts`.

[10] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, (2008), `https://bitcoin.org/bitcoin.pdf`.

[11] Leslie Lamport , Marshall Pease , Robert Shostak, *The Byzantine Generals Problem*, ACM Transactions on Programming Languages and Systems 4, **3** (1982), 382–401.

[12] Federal Information Processing Standard, *Descriptions of SHA-256, SHA-384 and SHA-512*, (2001), http://www.iwar.org.uk/comsec/resources/cipher/sha256-384-512.pdf.

[13] Josh Stark, *Making Sense of Blockchain Smart Contracts*, (2016), http://www.coindesk.com/making-sense-smart-contracts/.

[14] Scott Nadal Sunny King, *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*, (2012), https://peercoin.net/assets/paper/peercoin-paper.pdf.

[15] Pavel Vasin, *BlackCoin's Proof-of-Stake Protocol v2*, (2014), https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf.

Sažetak

U ovom radu opisana je blockchain tehnologija, kako s tehničkog tako i s praktičnog gledišta. Prvo poglavlje opisuje vrlo važnu funkciju koju koristi blockchain tehnologija, to je SHA-256 kriptografska hash funkcija. Drugo poglavlje daje definiciju blockchain tehnologije, prikazuje strukturu bloka i opisuje blockchain sustav građen prema decentraliziranom modelu ravnopravnih partnera . Uloga kriptografije i različiti algoritmi za postizanje konsenzusa u distribuiranom sustavu objašnjeni su u trećem poglavlju. Četvrto poglavlje daje pregled najvažnijih slučajeva upotrebe kao i aplikacija baziranih na blockchain tehnologiji naglašavajući njihov potencijal za buduću primjenu u raznim industrijama. Predstavljene su najpoznatije kriptovalute i njihove karakteristike. Također je objašnjeno što su to pametni ugovori i način na koji funkcionira blockchaina nove generacije na koji se zapisuju pametni ugovori. U sklopu rada izrađene su dvije jednostavne web aplikacije koje su opisane u zadnjem poglavlju.

Summary

In this thesis, blockchain technology is introduced both from a technological and functional point of view. First chapter describes a very important function which is used by blockchain technology, that is SHA-256 cryptographic hash function. Second chapter deals with definition of the blockchain and the structure of each block. It also describes decentralized peer-to-peer blockchain network. The role of cryptography and different consensus mechanisms for blockchain technologies are discussed in third chapter. Then in the fourth chapter we present some practical applications of the technology blockchain. In particular, we have concentrated on those applications which are currently identified as showing best promise for being applied in the industry. Cryptocurrencies features and next generation blockchains with smart contracts are explained too. Two simple web application have been developed as a part of this thesis and they are presented in last chapter.

Životopis

Domina Hozjan rođena je 7. listopada 1992. godine u Čakovcu. Školovanje je započela godine 1999. u osnovnoj školi u Ivanovcu. Nakon završetka, 2007. upisuje srednju školu Gimnaziju Josipa Slavenskog Čakovec, smjer prirodoslovno-matematička gimnazija. 2011. godine upisuje preddiplomski sveučilišni studij Matematika na Prirodoslovno-matematičkom fakultetu u Zagrebu. Nakon tri godine završava navedeni studij čime stječe akademski naziv Baccalaurea matematike (sveučilišna prvostupnica matematike). Po završetku preddiplomskog studija, godine 2014. upisuje diplomski studij Računarstvo i matematika na istom fakultetu u Zagrebu. Tijekom posljednje dvije godine studija radi u tvrtki Corvus info d.o.o. na poziciji programera ("junior software developer").